IBM

IBM Systems - iSeries

Systems management
Performance

Version 5 Release 4

IBM

IBM Systems - iSeries

Systems management
Performance

Version 5 Release 4

> **Note**
>
> Before using this information and the product it supports, read the information in "Notices," on page 147.

Contents

Performance

How much do you invest in managing the performance of your system? The needs of your business change, sometimes sooner than you expect.

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 on an iSeries™ server can be a complex task that requires a thorough understanding of iSeries work management. Understanding all the different processes that affect system performance can be challenging for the inexperienced user. Resolving performance problems requires the effective use of a large suite of tools, each with its own unique set of requirements and supported functions. Even after you have gathered and analyzed performance data, knowing what to do with that information can be daunting.

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 146.

Related concepts

Work management

What's new in V5R4

This topic describes what information is new or significantly changed in this release.

How to see what's new or changed

To help you see where technical changes have been made, this information uses:
- The ≫ image to mark where new or changed information begins.
- The ≪ image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to users.

The information on "Intelligent Agents" on page 67 is now included in the Performance PDF.

What's new: Monitors

Find out about the new and changed monitor functions.

Graph history
- You can now save a screen capture of the **Graph History** or the **Systems Monitor** window, or just the graph to your local drive. Select **File → Save Window As, File → Save Graph As**.
- You can print your graphs from the **Graph History** or the **Systems Monitor** window. You can print the entire window, or just the graph. Select **File → Print**.
- The System Monitor has the ability to link all of the displayed graphs together. When you select this option, all of the graphs that make up the monitor are coordinated in terms of the time-slice shown and the scroll position of the graph. If you scroll to one position in a graph, all of the graphs in the monitor simultaneously scroll to that same position. Select **View → Coordinate Graphs**.

- You can drag and drop the graphs in the **Systems Monitor** window into any visual configuration that you want, or resize the window. When you close the Systems Monitor window, the size and positions of the graphs are saved in that configuration the next time that you open the window.
- You can change the colors of the lines on the graph using a menu option on the **Graph History** or **Systems Monitor** window. Select **View → Set Monitor Colors**.
- The **Graph History** window now displays the graph history status. You also can re-create the graph history data if it is missing.

For more information, see "Graph history" on page 96.

System Monitor

Exclude heavily utilized communication lines, such as fax lines, from the system monitor graph. For example, when you have two communication line utilization metrics, the average of all of the communication lines is plotted. Thus, if there are one or more lines that you do not want to include in the average, such as a line that is heavily loaded because of fax traffic, you can optionally exclude these lines. For instructions on how to do this, use the online help for System Monitors.

From the **System Monitors Properties** window, click **Help**. From the **Help** window, click **Help Topics**. From the **Management Central Help Topics** window, click **Contents → How To → System Monitors → Excluding communication lines from a system monitor**.

What's new: CL commands and Collection Services

Read about the changes to the control language (CL) commands, performance database files, and Collection Services.

CL commands

You can use the Dump Main Memory Information (DMPMEMINF) command to dump main memory. You can analyze the output from this command to determine what is causing accumulation of temporary objects or database files.

You can use the Print Component Report (PRTCPTRPT) command to print information about Domino® servers.

The DSPACCGRP and ANZACCGRP commands are no longer supported.

Collection Services

Collection Services can now report CPU utilization data for up to 64 processors.

There have been changes in how the wait statistics are reported. For more information, see "Find wait statistics for a job, task, or thread" on page 64.

Cross-partition performance data

Disk data is now reported for all partitions. This information is stored in the QAPMLPAR file.

ARM performance data

You can use collection services to collect Application Response Measurement (ARM) performance data. This information is stored in the QAPMARMTRT and QAPMUSRTNS database files.

Performance database files

The following table shows the new and changed database files.

Database file	Description
QAPMARMTRT	This new database file reports data related to Application Response Measurement (ARM) transactions. ARM is a set of APIs that report the progress of application transactions.
QAPMASYN	This database file contains a new field: ASDUP.
QAPMCONF	This database file contains new record keys for reporting larger system auxiliary storage pool (ASP) capacity.
QAPMDISK	This database file contains larger fields for reporting unit capacity/space available for larger disk drives, and new fields for supporting storage adapters, which do not require input/output processors (IOP) to manage them.
QAPMIOPD	This database file contains a new field: XINWSH.
QAPMJOBMI, QAPMJOBS, and QAPMJOBL	These database files contain new fields for total accumulated CPU time, and for IP addresses that are associated with a thread.
QAPMJOBOS	This database file contains new fields for certain database operations.
QAPMLPAR	This database file contains several new fields for handling cross-partition data.
QAPMPOOLB	This database file contains a new field for the unallocated pool space.
QAPMSYSTEM, QAPMSYS, and QAPMSYSL	These database files contain several new and changed fields.
QAPMTCPIFC and QAPMTCP	The TCP/IP performance data reflects data for both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6).
QAPMUSRTNS	This database file now reports ARM transactions, as well as other transaction types.

Related reference

"Collect system performance data" on page 15
Collection Services regularly collects information about system performance. Often, analyzing performance data begins with this information.

What's new: Performance Tools licensed program

This topic highlights changes to the Performance Tools licensed program for V5R4.

Job trace reports

The following job trace reports have been changed to take advantage of the Job Trace functionality in the Start Trace (STRTC), End Trace (ENDTRC), and Print Trace (PTRTTRC) commands:
- Job Trace Information report (QPPTTRCD)
- Job Trace Analysis Summary report (QPPTTRC1)
- Job Trace Analysis I/O Summary Report (QPPTTRC2)

Component report - Domino server activity

The component report contains a new section on Domino server statistics.

Changes to Performance Tool reports

The following Performance Tool reports have changed fields:
- System Report - Storage Pool Utilization
- System Report - HTTP Server Summary
- System Report - Disk Utilization
- System Report - Workload
- Component Report - Component Interval Activity
- Component Report - Storage Pool Activity
- Component Report - HTTP Server Activity
- Pool Interval Report - Pool Activity
- Pool Interval Report - Subsystem Activity
- Resource Interval Report - Disk Utilization Summary

Performance Tools plug-in

The Performance Tools plug-in for iSeries Navigator contains the following enhancements:
- The **All Jobs** view of the **Display Performance Data** window contains two new columns that show the IP address most recently associated with the job and the remote port number.
- You can print graphs that help with analyzing performance in the Performance Tools graphical user interface (GUI).

DSPPFRDTA (Display Performance Data)

The Display Performance Data (DSPPFRDTA) command contains the following enhancements:
- The Display by Jobs and Display Job Detail screens show the IP address that is most recently associated with the job and the remote port number that the network connection uses.

WRKSYSACT changes

The WRKSYSACT command generates the QAITMON output file. The QAITMON file has the following changes in V5R4:
- The fields for individual CPU utilization have been removed.
- The following new fields have been added:
  - CPU Used and CPU Available
  - Minimum and Maximum Individual CPU Time Used

What's new: PM iSeries
Read about how PM eServer™ iSeries automatically collects data.

The data that PM iSeries collects has been updated. PM iSeries has been improved to collect data more efficiently.

What's new: Performance explorer
Read about the changes to the performance explorer database files.

| **ADDPEXDFN command**

| TRCTYPE(*HEAP) is a new trace type that selects all of the heap events from the STGEVT (storage
| events) parameter.

| **Migration of performance explorer database files**

| When you migrate to a new release of i5/OS™, if the system finds incompatible PEX database files, it
| moves these files to the QPEXD*vrmxx* library.

Printable PDF

Use this to view and print a PDF of this information.

To view or download the PDF version of the performance topic, select Performance (about 2187 KB). This
PDF does not include the performance database table information or the sample Performance Tools
reports.

To view or download the PDF version of the performance database table information, select Performance
database tables (about 2781 KB).

To view or download the PDF version of the Performance Tools report information, select Performance
Tools reports (about 1149 KB).

You can also view or download these related topics:
* Management Central (about 946 KB) includes information about how to set up your endpoint systems
  and system groups, as well as information about all the ways the Management Central function can
  help you streamline your server administration tasks, such as:
  – Manage users and groups
  – Package and send data
  – Run commands
* Work Management (about 2228 KB) describes these work management concepts:
  – Daily work management
  – The structure of your system
  – How work gets done
  – Schedule your tasks or jobs with Advanced Job Scheduler.
| * Schedule jobs with Advanced Job Scheduler includes information about managing jobs. For example,
| you can notify users if CPU utilization reaches a specified threshold.

Saving PDF files

To save a PDF on your workstation for viewing or printing:
1. Right-click the PDF in your browser (right-click the link above).
| 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 (www.adobe.com/products/acrobat/readstep.html) .

Plan for performance

Setting performance objectives for your iSeries server will allow you to have measurable performance benchmarks to compare your performance data. This topic will explain how to set those benchmarks and how to use them later.

Planning your system's performance requires you to set performance objectives, create benchmarks based on those objectives, and plan for your system's growth. This section guides you through the necessary steps in planning your system's performance.

When planning your system's performance, you will need to fully understand the business requirements your system is addressing and be able to translate those business needs into performance objectives. Keep in mind that as the business needs evolve, the performance objectives must also evolve.

Perhaps the best way to start is to estimate the maximum hourly and daily interactive transaction throughput required of your computer system during your peak business periods. After that, you can decide what average response time is acceptable to your local and remote workstations. You should think about how long your regular batch processes take, and how to schedule them so that they complete in time to achieve your business requirements.

You can then establish a base set of statistics, which should then be documented in a performance objective plan containing:
- The peak transactions per hour
- The peak transactions per day
- Acceptable average response time for local workstations
- Peak interactive transactions
- A list of the major scheduled batch jobs with times when they will be run and their expected duration
- A list of other unscheduled batch jobs that may be required

To plan for performance, complete the following tasks:

Set system benchmarks

Setting good system benchmarks will give you performance data for a properly tuned system. These performance benchmarks from both before and after system changes provide important information for both troubleshooting and planning.

You should establish system benchmarks before any major change in the system configuration, for example adding a new interactive application or performing a system upgrade. Maintaining accurate benchmark information can provide essential troubleshooting information. At a minimum, benchmarks should include current collection objects from Collection Services. Depending on your environment you may need to maintain more detailed information using Performance Explorer.

Setting up a benchmark requires:
- That the correct iSeries configuration is available
- That the application and the data are representative and valid
- That the correct version of all programs and software to be used are available
- That the required number of users and workstations are available to run the test
- That the transactions are well defined for each user

Running meaningful benchmarks for interactive workloads is almost impossible without special equipment that allows you to simulate a user at a workstation. To run a batch benchmark is, of course, not as complex a task as to test performance of interactive applications, and the first three points above

are still valid for this type of test. However, setting system benchmarks on concurrent batch and interactive work, which is frequently the actual customer environment, also requires the appropriate number of users and workstations.

IBM® developed a benchmark called the Three-in-One Benchmark to mirror the real-world demands facing IT companies. The Three-In-One Benchmark clearly demonstrates that the iSeries server is an excellent solution for today's small and medium businesses, which helps them run the applications they need without worrying about performance.

> **Related concepts**

> "Collection Services" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

> "Performance explorer" on page 128
> 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 information**

> Three-In-One Benchmark

Determine 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 eServer iSeries
> PM iSeries completely automates collecting data, analyzing data, and archiving data and provides you with summarized performance and capacity information that is easy to understand. PM iSeries helps you plan for and manage system resources through ongoing analysis of key performance indicators. This function ships with the i5/OS licensed program. There is nothing that you need to do other than activate the function and periodically check that the data is being collected and transmitted to IBM. All collection sites are network secure, and the PM iSeries service transmits only nonproprietary performance data to IBM. The time of the transfer is completely under your control.

Workload Estimator
> The Workload Estimator is a tool that helps you size your system needs based on estimated workloads for specific workload types. Through a web-based application, you can size the

upgrade to the required iSeries system that accommodates your existing system's utilization, performance, and growth as reported by PM iSeries. As an additional option, sizings can also include capacity for adding specific applications like Domino, Java™, and WebSphere®, or the consolidation of multiple workloads on one system. This capability allows you to plan for future system requirements based on existing utilization data coming from your own system.

PATROL for iSeries - Predict

This product helps manage iSeries performance by automating many of the routine administration tasks required for high availability and optimal performance. Additionally, it offers detailed capacity planning information to help you plan the growth of your iSeries environment.

See Select a performance management strategy for more information about creating and implementing a performance strategy.

Related concepts

"IBM Performance Management for eServer iSeries" on page 99
PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

"Workload Estimator for iSeries" on page 142
Helps you plan the size and timing requirements of your next upgrade. This tool is often used with PM iSeries to analyze trends in system performance and helps you efficiently manage the growth and expansion of your iSeries server.

"PATROL for iSeries (AS/400) - Predict" on page 143
PATROL for iSeries (AS/400®) - Predict helps manage iSeries performance by automating many of the routine administration tasks required for high availability and optimal performance. Additionally, this product offers detailed capacity planning information to help you plan the growth of your iSeries environment.

Related reference

"Select a performance management strategy"
Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.

Select a performance management strategy

Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.

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.

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 use PM iSeries 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 iSeries server needs an upgrade.

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

Collection Services
Collect sample data at user-defined intervals for later analysis.

Graph History
Display performance data collected with Collection Services.

PM iSeries
Automate the collection, archival, and analysis of system performance data.

Performance Tools
Gather, analyze, and maintain system performance information.

Monitors
Observe graphical representations of iSeries 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 PM iSeries.

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

Collection Services
Collect sample data at user-defined intervals for later analysis.

Graph History
Display performance data collected with Collection Services.

PM iSeries
Automate the collection, archival, and analysis of system performance data.

Performance Tools
Gather, analyze, and maintain system performance information.

Monitors
Observe graphical representations of iSeries system performance, and automate responses to predefined events or conditions.

Performance explorer
Collect detailed information about a specific application or system resource.

Large business

The large business has resources devoted to managing performance.

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

Collection Services
Collect sample data at user-defined intervals for later analysis.

Graph History

Display performance data collected with Collection Services.

PM iSeries
Automate the collection, archival, and analysis of system performance data.

Performance Tools
Gather, analyze, and maintain system performance information.

Monitors
Observe graphical representations of iSeries system performance, and automate responses to predefined events or conditions.

Performance explorer
Collect detailed information about a specific application or system resource.

iDoctor for iSeries
Analyze trace data to improve system and application performance.

Performance Trace Data Visualizer (PTDV)
View trace data from a Java application.

Related concepts

"Determine when and how to expand your system" on page 7
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.

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"Graph history" on page 96
Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

"IBM Performance Management for eServer iSeries" on page 99
PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

"Performance explorer" on page 128
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.

"iDoctor for iSeries" on page 137
The iDoctor for iSeries plugin consists of a variety of software tools for managing performance, for example, PEX Analyzer for detailed trace data analysis and Job Watcher for trace-level information about a job's behavior.

"Performance Trace Data Visualizer (PTDV)" on page 138
Performance Trace Data Visualizer for iSeries (PTDV) is a Java application that can be used for performance analysis of applications running on iSeries.

Related reference

"Performance Tools" on page 120
The Performance Tools licensed program includes many features to help you gather, analyze, and

maintain system performance information. This includes assistance in managing performance over a distributed network, collecting and reporting on both summary and trace data, and capacity planning.

"iSeries Navigator monitors" on page 87

Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

Set up your environment to manage performance

iSeries servers include powerful applications for managing system performance. However, they must be properly configured in order to meet the specific needs of your unique business environment. Learn how to configure applications to routinely collect, monitor, and analyze performance data.

The iSeries server includes several tools that regularly collect system performance data and monitor your system for performance trends and potential problems. Your unique requirements and environment will determine both the tools you choose to invest in and the configuration choices you should make. Effectively setting up your system will allow you to do accurate capacity planning as your system grows and to resolve performance problems when they occur.

Use the following topics to learn about and configure tools that will collect, monitor, and analyze your system performance.

Collection Services

Collection Services manages the routine collection of your system performance data. This tool regularly collects data and creates archives called collection objects. These collection objects may be accessed directly by some tools or converted into sets of database files for analysis with your own custom queries or by other tools and reports. Because Collection Services mainly provides data for other applications, the other tools you are using will significantly affect your configuration choices, including how frequently you collect data, the types of data you collect, and the length of time you will keep the data on your system.

PM iSeries

PM iSeries uses Collection Services to gather non-proprietary performance data, and sends it to IBM for storage and expert analysis. This eliminates the need to store and maintain it yourself. You can then access detailed reports and recommendations about your system's performance with a web browser.

iSeries Navigator monitors

The monitors included in iSeries Navigator use Collection Services data to track the elements of system performance of specific interest to you. Moreover, they can take specified actions when certain events, such as the percentage of CPU utilization or the status of a job, occur. Use this topic to learn how to use these monitors and how to set them up on your system.

Related concepts

"Collection Services" on page 30

Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"IBM Performance Management for eServer iSeries" on page 99

PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

Related reference

"iSeries Navigator monitors" on page 87

Monitors display current information about the performance of your systems. Additionally, you can

use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

Manage iSeries performance

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.

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.

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.

Track 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 for the iSeries server 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.

TheiSeries server includes several applications for tracking performance trends and maintaining a historical record of iSeries 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 which may 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:

Collection services
> Collection services gathers performance data at user-defined time intervals and then stores this information in collection objects on your system. Many of the other tools, including monitors, Graph history, PM iSeries, and many functions in the Performance Tools licensed program, rely on these collection objects for their data.

Graph history
> Graph history displays performance data collected with Collection Services over a specified

period of time through a graphical user interface (GUI). The length of time available for display depends on how long you are retaining the collection objects, and whether you are using PM iSeries.

PM iSeries

PM iSeries automates the collection, archival, and analysis of system performance data and returns clear reports to help you manage system resources and capacity.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"Graph history" on page 96
Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

"IBM Performance Management for eServer iSeries" on page 99
PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

Related reference

"Network performance" on page 25
The network design, hardware resources and traffic pressure often have a significant effect on the performance of e-business applications. You can use this topic for information on how to optimize network performance, and tune server communication resources.

Research 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. 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 system monitor 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 a performance explorer session, gather detailed information about that job's behavior on the server, and potentially make changes to the originating program.

Related tasks

"Basic performance tuning" on page 22
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.

Identify performance problem

Learn the common steps involved with identifying a performance problem.

When trying 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 i5/OS and the Performance Tools licensed program to measure the system performance.

Reviewing the measured data helps to further define the problem and helps 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.

Identify and resolve common performance problems

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

When performance problems occur on the iSeries server, 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. Many of these areas are available as system monitor metrics. However, there are several other ways to access information about them.

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.	• Work with Active Jobs (WRKACTJOB) command. • Work with System Activity (WRKSYSACT) command, which is part of Performance Tools licensed program. • The work management function in iSeries Navigator. • CPU utilization metrics within the iSeries Navigator system monitor. • Work with Disk Status (WRKDSKSTS) command.
Main storage	Investigate faulting and the wait-to-ineligible transitions.	• Disk storage metrics within the iSeries Navigator system monitor. • Work with System Status (WRKSYSSTS) command. • The Memory Pools function under Work Management in iSeries Navigator.

Area	Description	Available tools
Disk	Determine if there are too few arms or if the arms are too slow.	• Work with Disk Status (WRKDSKSTS) command. • Disk arm utilization metrics within the iSeries Navigator system monitor. • Performance Tools System and Component report.
Communications	Find slow lines, errors on the line, or too many users for the line.	• Performance Tools Component Report. • LAN utilization metrics within the iSeries Navigator system monitor.
IOPs	Determine if any IOPs are not balanced or if there are not enough IOPs.	• Performance Tools Component Report. • IOP utilization metrics within the iSeries Navigator system monitor.
Software	Investigate locks and mutual exclusions (mutexes).	• Performance Tools Locks report. • Performance Tools Trace report. • Work with Object Locks (WRKOBJLCK) command. • In iSeries Navigator, right-click on the suspect job under Work Management, and select **Details → Locked Objects**. • WRKSYSACT Option 6. • DSPPFRDTA. F6, Option 6.
Backup and recovery	Investigate areas that affect backup and recovery and save and restore operations.	• iSeries Performance Capabilities Reference (Save/Restore Performance chapter). • Why does my backup take so long after I restart my server? • Why do my backups take longer after I upgrade to a new release? • Why do my backups take longer after I change hardware on my server?

Related concepts

Work management function in iSeries Navigator

Related reference

"Monitor metrics" on page 90
To effectively monitor system performance, you must decide which aspects of system performance you want to monitor. Management Central offers a variety of performance measurements, known as **metrics**, to help you pinpoint different aspects of system performance.

Frequently asked questions

Related information

iSeries Performance Capabilities Reference (Save/Restore Performance chapter)

Collect system performance data

Collection Services regularly collects information about system performance. Often, analyzing performance data begins with this information.

Collecting data is an important step toward improving performance. When you collect performance data, you gather information about your server that can be used to understand response times and throughput. It is a way to capture the performance status of the server, or set of servers, 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.

You can use Collection Services to collect performance data, create performance files with the Create Performance Data (CRTPFRDTA) command, convert them to current release with Convert Performance Data (CVTPFRDTA) command or through the Performance Tools plugin in iSeries Navigator, and then create reports or create your own queries by using the information in the performance database files.

For more information about performance data, see the following:

Collection Services
> See how to collect performance data for analysis and how to customize your collections.

Performance database files
> Find an overview of the performance database files that are available to you and see detailed field data for each performance database file.

In addition, you can use either the Performance Management APIs or the performance collection CL commands to start, end, and cycle collections, as well as to change and retrieve system parameters for the data collected.

> **Related concepts**
>
> "Collection Services" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)
>
> "What's new: CL commands and Collection Services" on page 2
> Read about the changes to the control language (CL) commands, performance database files, and Collection Services.
>
> **Related reference**
>
> Performance Management APIs

Collect 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 iSeries server 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:

iSeries Navigator monitors
> The monitors included in iSeries Navigator provide current and recent data on a wide variety of metrics. Additionally, you can configure them to take a specified action when certain events occur.

i5/OS performance commands

i5/OS includes several important functions to help you manage and maintain system performance.

PM iSeries

PM iSeries uses Collection Services to gather non-proprietary performance data and sends it to IBM for storage and expert analysis. This eliminates the need to store and maintain it yourself. You can then access detailed reports and recommendations about your system's performance and trend analysis with a web browser.

Related concepts

"IBM Performance Management for eServer iSeries" on page 99
PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

Related reference

"iSeries Navigator monitors" on page 87
Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

"Commands for i5/OS performance" on page 139
i5/OS includes several important functions to help you manage and maintain system performance.

Collect information about an application's performance

An application might be performing slowly for a variety of reasons. You can use several of the tools included in i5/OS 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, Performance Trace Data Visualizer (PTDV), and iDoctor. Alternately, you can get an overview of application performance by using the Job monitor to track individual server performance and Performance Tools 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 you have tried all other collection options.

Performance Explorer

This tool 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.

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

Performance Trace Data Visualizer for iSeries (PTDV)

This tool is a Java application that can be used for performance analysis of applications running on iSeries. PTDV works with the performance explorer function of i5/OS to allow the analyst to view program flows and get details (such as CPU time, current system time, number of cycles, and number of instructions) summarized by trace, job, thread, and procedures. When visualizing Java application traces, additional details such as the number and type of objects created, as well as information about Java locking behavior, can be displayed. There is also support for performance explorer events generated by the WebSphere Application Server. PTDV allows sorting of columns, exporting of data, and many levels of data summarization.

iDoctor for iSeries

> The PEX Analyzer function in iDoctor includes a software tool specifically geared towards analyzing trace data to improve system and application performance. This detailed analysis gives a low-level summary of disk operations, CPU utilization, file-open operations, machine interface (MI) programs, wait states, disk space consumption, and much more. The client component is an iSeries Navigator plugin that allows a user to condense and display iSeries trace data graphically.

Start Performance Trace (STRPFRTRC) command

> i5/OS includes a command to collect multi-programming and transaction data. This command collects the data that STRPFRMON collected in previous releases. After running this command, you can export the data to a database file with the Dump trace (DMPTRC) command.

Related concepts

"Performance explorer" on page 128
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 Trace Data Visualizer (PTDV)" on page 138
Performance Trace Data Visualizer for iSeries (PTDV) is a Java application that can be used for performance analysis of applications running on iSeries.

"iDoctor for iSeries" on page 137
The iDoctor for iSeries plugin consists of a variety of software tools for managing performance, for example, PEX Analyzer for detailed trace data analysis and Job Watcher for trace-level information about a job's behavior.

Related reference

"iSeries Navigator monitors" on page 87
Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

"Performance Tools" on page 120
The Performance Tools licensed program includes many features to help you gather, analyze, and maintain system performance information. This includes assistance in managing performance over a distributed network, collecting and reporting on both summary and trace data, and capacity planning.

Start Performance Trace (STRPFRTRC) command

Related information

Performance Trace Data Visualizer

Dump trace data:

Deciding when to dump trace data is a significant decision because the dump affects system performance.

The Dump Trace (DMPTRC) command puts information from an internal trace table into a database file. It is not good 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" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)
>
> **Related reference**
>
> Dump Trace (DMPTRC) command

Dump 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
```

Scenario: Improve 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 iSeries server 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 i5/OS and Performance Tools licensed program (5722-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 iSeries Navigator system 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 IOP utilization is greater than 60%, add an additional IOP and assign some disk resource to it.
- If the page faults in the user pool are unacceptably high, refer to the topic 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.

4. Run iDoctor with the **Task switch** option for five minutes. Then analyze the resulting trace data with the task switch monitor. Identify and resolve any of the following:
   - Jobs waiting for CPU
   - Jobs faulting
   - Seize conflicts

   Related concepts

   "Automatically tune performance" on page 23
   Most users should set up the system to make performance adjustment automatically. When new systems are shipped, they are configured to adjust automatically.

   "iDoctor for iSeries" on page 137
   The iDoctor for iSeries plugin consists of a variety of software tools for managing performance, for example, PEX Analyzer for detailed trace data analysis and Job Watcher for trace-level information about a job's behavior.

   Related reference

   STRPFRTRC command

   Related information

   iSeries Performance Capabilities Reference PDF

Display 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.

Display near real-time performance data

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

i5/OS commands
> There are many commands in the base operating system that will let you view current information about specific areas of system performance.

Performance Tools plug-in
> The Performance Tools licensed program includes a plug-in for iSeries Navigator that displays performance data from Collection Services collection objects. You can also view detailed information about the jobs on the system and print Performance Tools reports.

System and Job monitor

> These monitors display performance data for many system elements. Monitor data is based on the collection objects, and will display data as it is collected, according to the collection interval in Collection Services.

Display historical performance data

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

PM iSeries

> PM iSeries automates the collection, archival, and analysis of system performance data and returns clear reports to help you manage system resources and capacity.

Graph history

> Graph history provides a graphical display of up to a week's worth of performance data depending on the retention period in Collection Services. With PM iSeries, Graph History can display much longer periods of data collection.

> **Related concepts**

> "Collection Services" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

> "IBM Performance Management for eServer iSeries" on page 99
> PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

> "Graph history" on page 96
> Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

> **Related tasks**

> "Performance Tools plug-in" on page 123
> You can view system resource utilization data in iSeries Navigator. You can view the data, graph it, and summarize it into reports. Find information about how to access this function here.

> **Related reference**

> "Commands for i5/OS performance" on page 139
> i5/OS includes several important functions to help you manage and maintain system performance.

> "iSeries Navigator monitors" on page 87
> Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

Tune 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 allow your servers to make the best use of the system resources, and to allow workloads to run as efficiently as possible. 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.

IBM also offers a tool that allows you to improve both the I/O subsystem and system response times by reducing the number of physical I/O requests that are read from disk. Learn how you can improve your system performance with Extended Adaptive Cache.

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 concepts

"Extended Adaptive Cache" on page 140
You can use Extended Adaptive Cache to improve system performance by collecting disk usage data, and then using those statistics to create a cache, effectively reducing the physical I/O requests for the disk.

Related reference

Thread affinity system value

Processor multitasking system value

Basic performance 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 are 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, and WRKACTJOB commands to display performance data. You can also use the Performance Tools command, Work with System Activity (WRKSYSACT), 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 will need to identify the source of the performance problem and to make specific corrections.

> **Related reference**
>
> "Research a performance problem" on page 13
> 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.

Automatically tune performance

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), select **Configuration and Service** → **System Values** → **Performance in iSeries Navigator**. 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, select **Configuration and Service** → **System Values** → **Performance in iSeries Navigator**. 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), select **Configuration and Service** → **System Values** → **Performance in iSeries Navigator**. 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 information**
>
> "Scenario: Improve system performance after an upgrade or migration" on page 19
> 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.
>
> Memory Pools

Determine 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. Find out more about this concept.

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 will vary. IBM eServer 5*xx* 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 *iSeries 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

QPRCMLTTSK system value

Related information

iSeries Performance Capabilities Reference PDF

Manage e-business performance

Managing performance in an e-business environment introduces several new problems for the i5/OS administrator.

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

Related concepts

Work management

Programming: Java

HTTP server

Related information

Domino for iSeries sizing and performance tuning

Client performance

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

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

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

- Have adequate memory. Resource intensive applets, and interfaces using complex forms and graphics may also place demands on the client's processor.
- Utilize 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 e-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:

Collection Services
> Collection Services collects performance data for communication resources at regular intervals. Of particular interest, the performance data files QAPMTCP and QAPMTCPIFC store information about TCP servers. You can reference this data by querying the files directly, or by using the reports included in the Performance Tools licensed program.

System monitor
> You can use the system monitors to provide information about how system resources, including communications hardware, are being used on a server. In particular, the line utilization and IOP metrics within the system monitor can provide valuable data about network performance.

Track performance
> Several applications and tools allow you to routinely collect performance data for communication resources on the server and to monitor their performance over time.

iSeries Performance Capabilities Reference
> 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.

iSeriesNetwork.com
> This web site hosts many resources for optimizing your network plan and resources. In particular, refer to the articles "Cultivate your AS/400 Networks" and "8 tools for better network performance."

> **Related concepts**

> "Collection Services" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

> "Track performance" on page 12
> 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**

> "iSeries Navigator monitors" on page 87
> Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

> **Related information**

> iSeries Performance Capabilities Reference PDF

> iSeries Network.com

Java performance in i5/OS

i5/OS provides several configuration options and resources for optimizing the performance of Java applications or services on the iSeries server. 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 i5/OS environment and of the Java application, to get optimal performance.

Use the following resources to learn about the Java environment in i5/OS and the available tips and tools for analyzing and improving Java performance.

Java performance
> There are several important configuration choices and tools to help you get the best performance from Java-based applications.

Collect information about an application's performance
> There are several tools available to help you monitor and tune an applications performance in i5/OS. Use this topic to learn how to use performance traces, performance explorer (PEX), and similar tools, to help you measure and improve application performance.

iSeries Performance Capabilities Reference
> 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 servers
> Use this Redbook 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 server
> This Redbook provides in introduction to J2EE, and offers suggestions and examples to help you successfully implement J2EE applications on the server.

> **Related concepts**

> Programming: Java

IBM HTTP Server performance

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

IBM HTTP server for iSeries can play an important role in the end-to-end performance of your Web-based applications, and several new improvements allow you to effectively monitor and improve Web server performance. In particular the new Fast Response Caching Accelerator (FRCA) may allow you to significantly improve HTTP server performance, particularly in predominantly static environments.

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

Collection Services
> You can use Collection Services to collect HTTP server performance data and monitor the results over time. The performance data files QAPMHTTPB and QAPMHTTPD store HTTP server data for each collection interval. QAPMHTTPB provides basic information, while QAPMHTTPD provides more detailed statistics. You can query these data files directly, or refer to the System and Component reports in the Performance Tools Licensed Program.

IBM HTTP Server for iSeries
> Refer to this topic for information on setting up, configuring, and managing an HTTP server on iSeries servers. This topic also includes descriptions of the latest enhancements, like the Fast Response Caching Accelerator (FRCA), to this product.

iSeries Performance Capabilities Reference
> The Performance Capabilities Reference provides detailed information, reports, and examples that

can help you configure or tune your iSeries 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 servers
Use this Redbook to get an in-depth description of HTTP Server (Powered by Apache) on i5/OS, including examples for configuring HTTP Server in common usage scenarios.

AS/400 HTTP Server Performance and Capacity Planning
Use this Redbook to learn about HTTP server impacts on performance tuning and planning. This publication also includes suggestions for using iSeries performance management tools to collect, interpret, and respond to Web server performance data.

System Report - HTTP Server Summary
Use this report to evaluate iSeries systems that are used for Web serving applications.

Related information

Performance data files: QAPMHTTPB

Performance data files: QAPMHTTPD

WebSphere performance

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

Managing iSeries server performance in a WebSphere environment presents several challenges to the iSeries 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 adjust server resources in a WebSphere environment.

WebSphere Application Server performance considerations
This web site provides resources for each version of WebSphere Application Server on the iSeries server, including many useful performance tips and recommendations. This resource is particularly valuable for environments using servlets, Java Server Pages (JSPs) and Enterprise Java Beans (EJBs).

DB2® UDB/WebSphere Performance Tuning Guide
This Redbook 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 Servers
Use this Redbook 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.

WebSphere V3 Performance Tuning Guide
This Redbook offers detailed recommendations and examples for optimizing WebSphere V3 performance on the iSeries server.

iSeries Performance Capabilities Reference
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 WebSphere specific performance tips.

Related concepts

e-business and Web serving

Applications for performance management

Managing performance on iSeries systems requires the use of a variety of specialized applications. Each of these applications offers a specific insight into system performance. This topic explains several applications and the intended use of each application.

Many of the applications for performance management have several functions. Knowing exactly which component of the available suite of applications best suits a given situation can be complex. The following topics provide detailed information about each of the performance management applications, including selection, use, and configuration.

As shown in the following figure, basically there are two performance collection functions on the iSeries servers:

- Collection Services, which collects interval data at the system and job level. You can run this continuously to know what is happening with your system. The interval data that is collected is either application defined or user defined.
- Performance explorer, which collects detailed data at the program and application level. It also traces the flow of work in an application and can be used to diagnose difficult performance problems. The data that is collected is by application-defined performance explorer trace points, such as with Domino, NetServer™, or WebSphere.

Both collection functions deposit their data into management collection objects. You can convert the data from the management collection objects by using the Create Performance Data (CRTPFRDTA) command for Collection Services data or by using the Create Performance Explorer Data (CRTPEXDTA) command for the performance explorer data.

This topic introduces the performance management applications that are available to work with either the Collection Services data or the performance explorer data.

The diagram shows the following labeled boxes and their relationships:

- Collection Services database (cylinder)
- Collection Services-based tools
- Performance Explorer-based tools
- Performance explorer database (cylinder)
- Management collection object (left)
- Performance Tools, Monitors, PM eServer iSeries, Workload Estimator
- iDoctor (PEX Analyzer), PTDV
- Management collection object (right)
- Collection Services: Interval data at system and job level
- Performance explorer: Detailed data at program and application level
- Application-defined and user-defined interval data
- Performance data
- Application-defined trace points
- iDoctor (Job Watcher), Heap Analyzer for Java Tools — Other performance tools

Collection Services

Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

Collection Services collects data that identifies the relative amount of system resource used by different areas of your system. Use Collection Services to:

- Easily manage your collection objects
- Collect performance data continuously and automatically with minimal system overhead
- Control what data is collected and how the data is used
- Move performance data between releases without converting the data
- Create performance data files that are used by Performance Tools
- Integrate your own programs to collect user-defined performance data into Collection Services.

How Collection Services works

Collection Services replaces the i5/OS performance monitor, which was called by the Start Performance Monitor (STRPFRMON) command. The Performance Monitor (STRPFRMON command) has not been available since V4R5. When you used the i5/OS performance monitor, your data was collected into as many as 30 database files.

Collection Services capabilities introduce a new process of collecting performance data. Collection Services stores your data for each collection in a single collection object, from which you can create as many different sets of database files as you need. This means a lower system overhead when collecting performance data. Even if you elect to create the database files during collection, you still experience a performance advantage over the i5/OS performance monitor because Collection Services uses a lower priority (50) batch job to update these files. The reduction in collection overhead makes it practical to collect performance data in greater detail and at shorter 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.
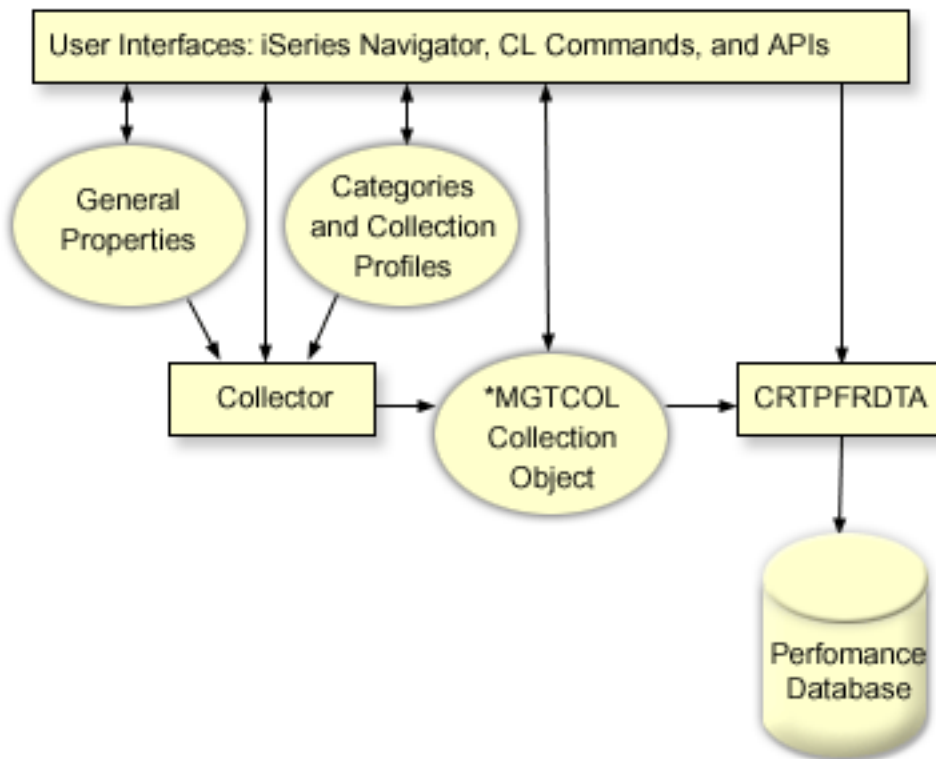
Collection Services allows you to gather performance data with little or no observable impact on system performance. You can use iSeries Navigator to configure Collection Services to collect the data you want as frequently as you want to gather it. A collection object, *MGTCOL, serves as an efficient storage medium to hold large quantities of performance data. Once you have configured and started Collection Services, performance data is continuously collected. When you need to work with performance data, you can copy the data you need into a set of performance database files.

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

User interfaces

    Several methods exist that allow you to access the different elements of Collection Services. For example, you can use CL commands, APIs, and theiSeries Navigator 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, *MGTCOL, serves as an efficient storage medium for large quantities of performance data.

Create Performance Data (CRTPFRDTA) 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, trace data files.

For an illustration of how Collection Services and system and job monitors work together on the system, refer to System and Job monitor interaction with Collection Services.

How to start Collection Services

You can start Collection Services by using any of the following methods. However, the information in the Performance topic focuses on iSeries Navigator methods.

Starting method	Description
Start Performance Collection (STRPFRCOL) command	Use the STRPFRCOL command to start the system-level collection of performance data by Collection Services.
iSeries Navigator	Perform a variety of Collection Services tasks by using iSeries Navigator. The table that follows below lists these tasks and links you to information about how to complete them.
Performance Management APIs	Use Performance Management APIs to start, customize, end, and cycle collections. In addition, you can use the APIs to work with the management collection objects or define your own transactions.
Traditional menu options	Type **GO PERFORM** in the character-based interface and select option 2 (Collect performance data) from the Performance Tools main menu. For additional information, go to the Performance Tools book.
PM iSeries	PM iSeries automates the start of Collection Services and then creates the database files during collection.

Related concepts

"Set system benchmarks" on page 6
Setting good system benchmarks will give you performance data for a properly tuned system. These performance benchmarks from both before and after system changes provide important information for both troubleshooting and planning.

"Track performance" on page 12
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.

"Dump trace data" on page 18
Deciding when to dump trace data is a significant decision because the dump affects system performance.

iSeries Navigator

"Time zone considerations for Collection Services" on page 40
When you review and analyze performance data, the actual local time of the collection can be significant.

"Monitor concepts" on page 89
Monitors can display real-time performance data. Additionally, they can continually monitor your system in order to run a selected command when a specified threshold is reached. Learn how monitors work, what they can monitor, and how they can respond to a given performance situation.

"Graph history" on page 96
Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

"Graph history concepts" on page 97
Contains a description of the available options for managing and displaying records of performance data.

"Data collection considerations for PM iSeries" on page 101
PM iSeries uses Collection Services to gather performance data. Learn how PM iSeries and Collection Services work together to provide the data you need.

"PM iSeries reports" on page 120
The iSeries server can be configured to send Collection Services data directly to IBM with PM iSeries. IBM then generates several reports that you can either view on the Web or have sent directly to you. Activating PM iSeries to automatically generate your reports not only saves you time and resources, but also allows you to plan ahead by forecasting your future growth needs.

"Performance Tools concepts" on page 121
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.

"Performance explorer concepts" on page 129
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 tasks

"Activate PM iSeries" on page 102
PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

"Customize data collections" on page 39
When you use Collection Services to collect performance data, you control what data is collected and how often it is collected.

"Use graph history" on page 98
This topic contains step-by-step instructions to view graph history through iSeries Navigator.

Related reference

Related information

Performance data files: QAPMDISK

Performance data files: QAPMIOPD

Performance data files: QAPMJOBMI

Performance data files: QAPMJOBS and QAPMJOBL

Performance data files: QAPMJSUM

Performance data files: QAPMLPAR

Performance data files: QAPMSYS and QAPMSYSL

Performance data files: QAPMSYSCPU

Performance data files: QAPMSYSTEM

System and job monitor interaction with Collection Services

Collection services is both a valuable tool for performance analysis as a stand alone application and as a
utility used by other applications for the collection of performance data.

Sometimes, performance analysis causes confusion when trying to determine which application is
responsible for activity you may see on your system. One easy rule to remember for this issue is that
even if it looks like those other applications are busy, there is one and only one data collection occurring
on the system at any given time.

The following scenarios explain the different combinations between system monitors and job monitors
and Collection Services and what Collection Services displays.

Collection Services is collecting data using the default values

In this scenario, there are no system monitors or job monitors active on the system. When viewing the Collection Services properties page and the *MGTCOL object properties view, you see something similar to the following:
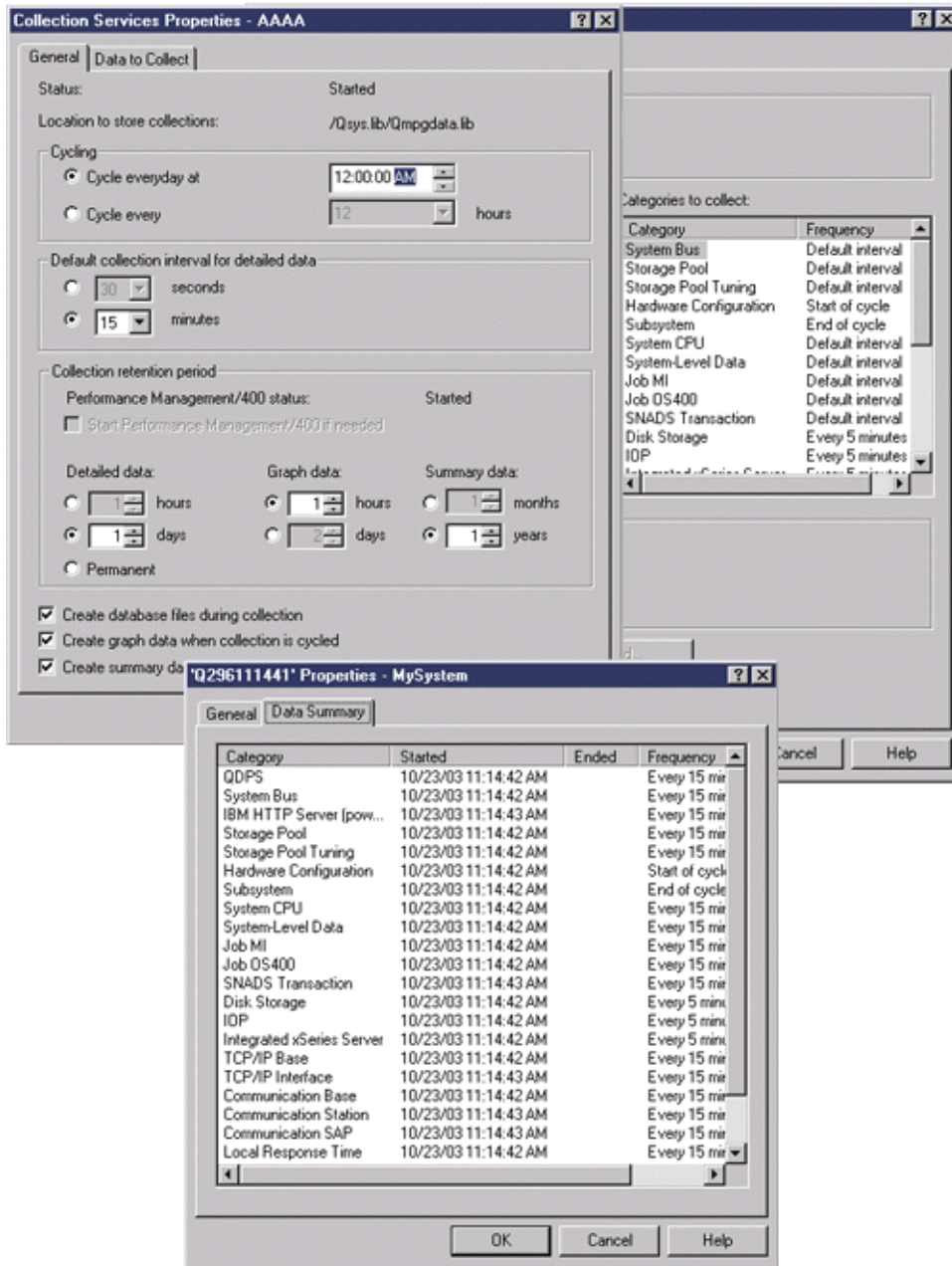
Both Collection Services and a system monitor are started

This scenario shows that Collection Services had already started at some point, and later someone started a system monitor to collect CPU Utilization (Average) metric data at 30-second intervals. Notice in the *MGTCOL object properties view that the collection interval for System Level Data, Job MI Data, and Job OS Data categories changed from 15 minutes to 30 seconds. This demonstrates that the same *MGTCOL object is being used, and only those categories necessary to calculate information for a given metric were changed to collect at the new interval.
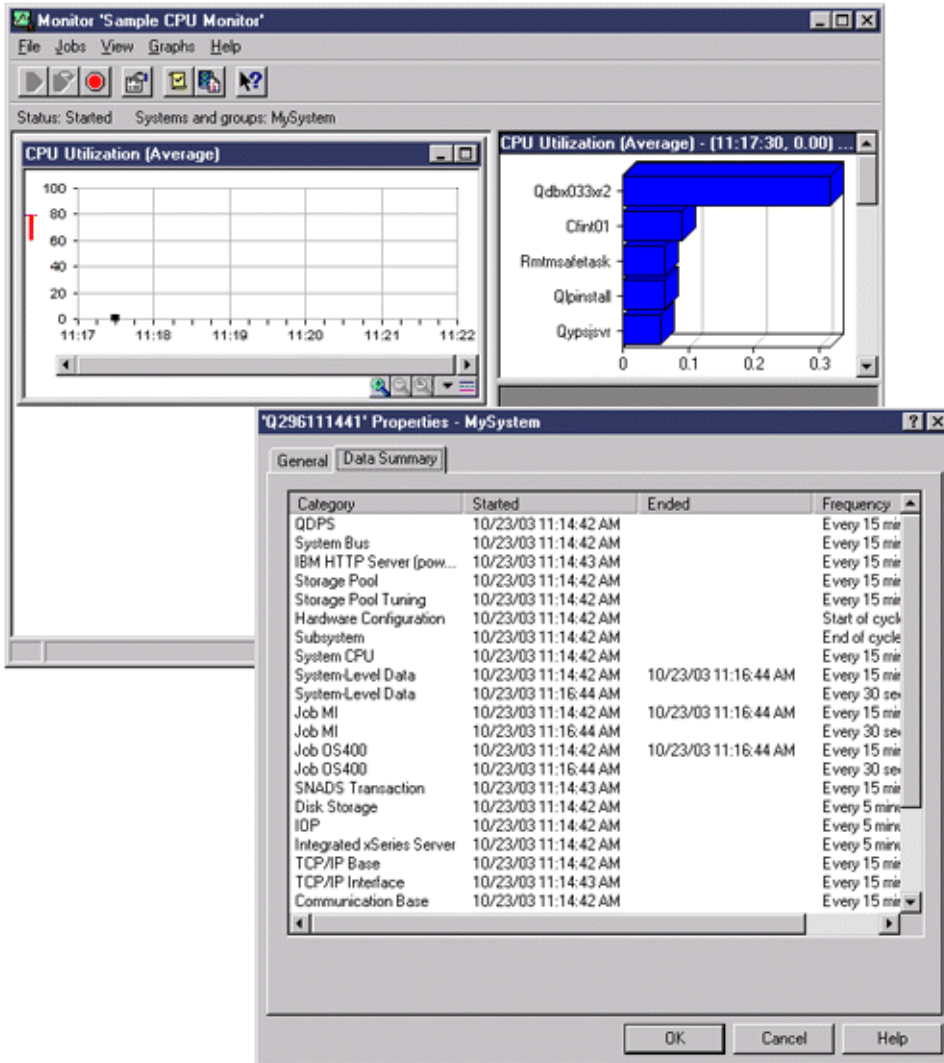
Collection Services stopped and system monitor remains started

In this scenario, Collection Services was stopped and the system monitor remains started and continues to collect data necessary to calculate the graph metrics.

Observe the following:

- The Collection Services properties page shows a status of **System collection stopped. Collecting for applications only**.
- The *MGTCOL object properties page shows that data collection has ended for all categories except for those necessary to calculate the graph metric data.
- The Collection Services list view shows the *MGTCOL object with a status of **Collecting...** This might be confusing; therefore, to get the status of Collection Services, look at the Collection Services Properties page.
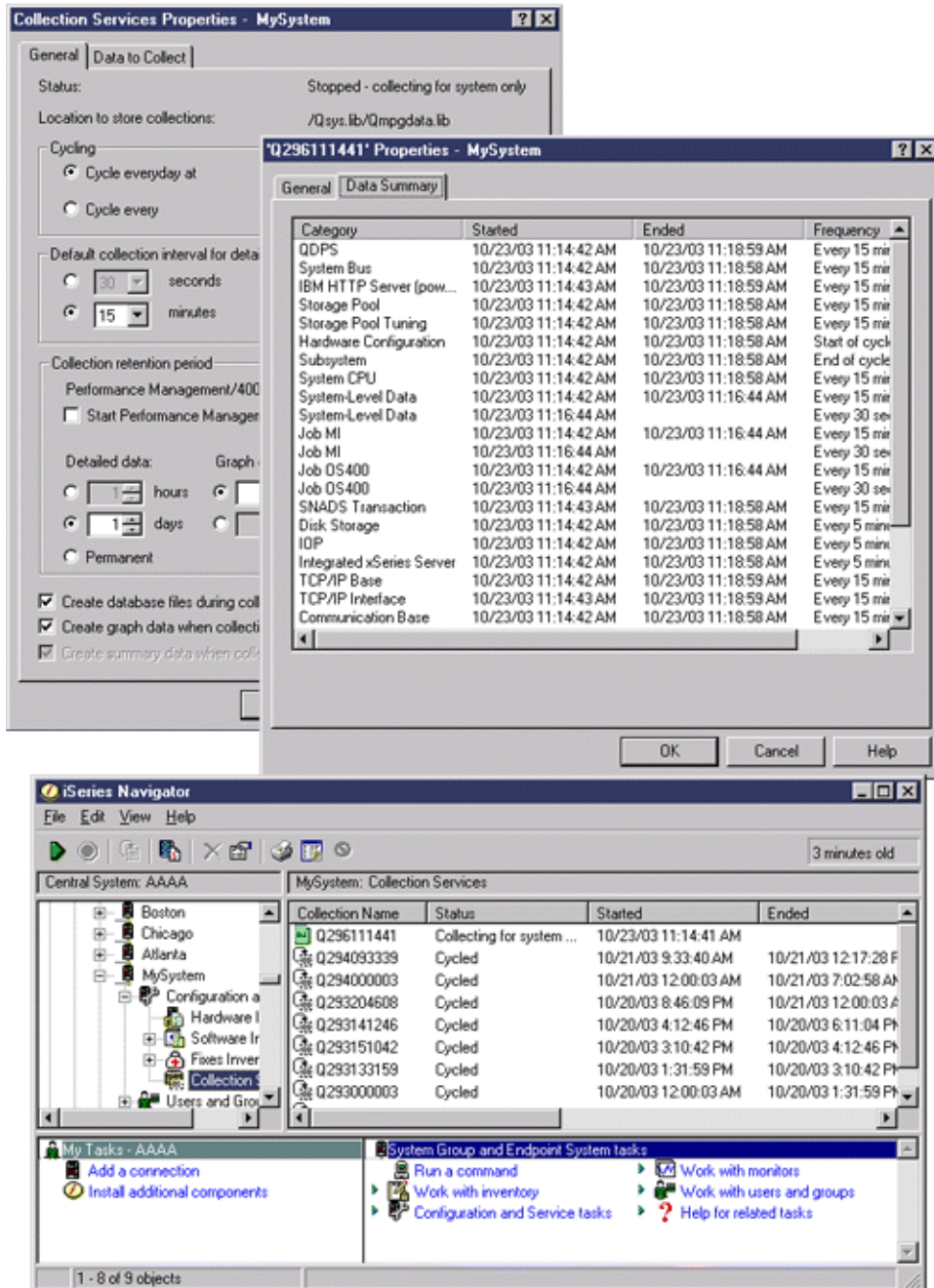
Create database files from Collection Services data

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, simply 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 iSeries Navigator interface or the CRTPFRDTA command.
- You can activate PM i5/OS, which automates the start of Collection Services and then creates the database files during collection.

You can use the database files that you have created with the Performance Tools for iSeries Navigator licensed program 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 run the Performance Tools reports. This action allows you to analyze the performance data on another system without affecting the performance of the source system.

Storing data in management collection objects instead of in database files

Why should you store the data 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.

Export the collected data

To export performance data from a management collection object to database files, follow these steps:
1. In iSeries Navigator, either select an endpoint system under **Management Central** or select a system to which you have a direct connection under **My Connections** (or your active environment).
2. Expand **Configuration and Service**.
3. Click **Collection Services**.
4. Right-click the management collection object that you want to export to database files and select **Create Database Files**.
5. On the **Create Database Files** dialog, select the categories from the collection object to include in the database files. You can also select a different time period and sampling interval, as long as the collection object contains data to support your selections.
6. Click **OK**.
   Related tasks

"Manage collection objects" on page 51
When you use Collection Services to collect performance data, each collection is stored in a single object.

Related reference

Create Performance Data (CRTPFRDTA) command

Create database files from an existing collection object:

You can export performance data from an existing management collection object to database files.

Follow these steps:

1. Expand **Configuration and Service** for the system from which performance data is being collected.
2. Select **Collection Services**.
3. Right-click the management collection object from which you want to export data to the database files.
4. You can first select **Properties** to display the characteristics of the data in the collection object. On the Data properties page, you can see the categories of data collected in this collection object as well as the intervals at which they were collected. You can use this information in selecting the data that you want to export. When you have reviewed this information, click **OK**.
5. Right-click the management collection object again and select **Create Database Files**. Complete the fields using the online help.
6. Click **OK**.

After you convert the data in the database files, you can use the Performance Tools for iSeries Navigator licensed program or other applications to produce performance reports.

Related reference

"Performance Tools" on page 120
The Performance Tools licensed program includes many features to help you gather, analyze, and maintain system performance information. This includes assistance in managing performance over a distributed network, collecting and reporting on both summary and trace data, and capacity planning.

Customize 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 corresponds to the settings for system data in the i5/OS performance monitor function that was provided by the Start Performance Monitor (STRPFRMON) command in previous releases. The **Standard plus protocol** profile corresponds to the STRPFRMON command settings for all data. Or you can select **Custom** to create your own customized profile. There are also several other profiles available, refer to the online help for detailed descriptions. For your customized profile, you can select from a list of available data categories, such as System CPU, Local Response Time, Disk Storage, and IOPs (input/output processors).

For each category of data that you collect, you can specify how often the data will be collected. For many categories, you will want to select the default collection interval, which you can set from predefined settings between 15 seconds and 60 minutes. (The recommended setting is 15 minutes.)

Note: When the default value is set to any specified time, all categories except those categories with explicit time intervals, such as, disk storage, input/output processors, and communications-related categories, use the specified time.

The collected data is stored in a management collection object (type *MGTCOL) called a collection. 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 interval from one hour to 24 hours, depending on how you plan to use the data.

To customize Collection Services on a system, follow these steps:

1. In iSeries Navigator, select either an endpoint system under **Management Central** or a system to which you have a direct connection under **My Connections** (or your active environment).
2. Expand **Configuration and Service**.
3. Right-click **Collection Services** and select **Properties**.
4. On the **General** page, you may want to specify a retention period longer than the default of 1 day. Collection Services may delete management collection objects and the data they contain from the system at any time after the retention period has expired. When the management collection object is created, an expiration date is assigned to it. Even if you move the collection object to another library, Collection Services will delete the object after it expires. You can specify **Permanent** if you do not want Collection Services to assign an expiration date to new collection objects. You will then have to delete these collection objects manually. To view the Graph History window, you must specify a Collection retention period of either Graph or Summary. When you specify these options, you can take advantage of the historical reporting capabilities, which allow you to do metric comparisons for multiple systems over extended periods of time. You can also specify the path of the location where you want to store your collections, how often you want to cycle collections, and the default collection interval. You can select to create database files automatically during collection.
5. Click the **Data to Collect** tab.
6. For **Collection profile to use**, select **Custom**. You can specify the collection interval for each category you select for your customized list.
7. Click **OK** to save your customized values.

Once you have customized Collection Services to the settings you prefer, you can right-click **Collection Services** again and select **Start Collection Services** to begin collecting performance data.

> **Related concepts**
>
> "Collection Services" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)
>
> "Graph history" on page 96
> Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

Time zone considerations for Collection Services:

When you review and analyze performance data, the actual local time of the collection can be significant.

For example, you may need to be sure which data was collected during the busiest period of the day so that it represents the heaviest workload experienced by the system under review. If some of the systems from which you collect performance data are located in different time zones, you should be aware of these considerations:

- When you start Collection Services for a system group, you start Collection Services at the same time on all systems in the group. Any differences in system time and date settings due to some systems being located in different time zones are not taken into account.
- If you start Collection Services with the Management Central scheduler, the time at which the scheduler starts the task is based on the system time and date of your central system in Management Central.
- The management collection objects for each endpoint system reflect start and end times based on the QTIME and QUTCOFFSET (coordinated universal time offset) system values of that endpoint system

and your central system. If the endpoint system is in a different time zone from your central system, and these system values are correctly set on both systems, the start and end times reported for collection objects are the actual times on the endpoint system. In other words, the start and end times reflect the value of QTIME on the endpoint system as it was at the actual point in time when those events occurred.

- The scheduling of a performance collection can cross a boundary from standard time to daylight saving time or from daylight saving time to standard time. If so, this time difference should be taken into account when scheduling the start time. Otherwise, the actual start and end times can be an hour later or earlier than expected. In addition, the reported start and end times for management collection objects are affected by this difference unless the QUTCOFFSET system value is adjusted each time the change to and from daylight saving time takes effect.

  Related concepts

  "Collection Services" on page 30
  Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)
  QTIME
  QUTCOFFSET (coordinated universal time offset)

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.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

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

> **Related reference**
>
> QypsRegCollectorDataCategory API
>
> QypsDeregCollectorDataCategory API
>
> QpmRtvActiveMgtcolName API
>
> QpmRtvMgtcolAttrs API
>
> QpmOpenMgtcol API
>
> QpmCloseMgtcol API
>
> QpmOpenMgtcolRepo API
>
> QpmCloseMgtcolRepo API
>
> QpmReadMgtcolData API
>
> QypsAddCollectorNotification API
>
> QypsRmvCollectorNotification API

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 146.

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.
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.

Related reference

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

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.

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.

Example: data collection program:

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

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

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" on page 42
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 146.

C++ sample code

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


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,
                    "12345678901234567890123456789012345678901234567890", 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 QIBM/ProdData/OS400/CollectionServices/lib.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

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 */
```

Manage collection objects

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

You can see a summary of the data in any management collection object by following these steps:

1. In iSeries Navigatorr, select either an endpoint system under **Management Central** or a system to which you have a direct connection under **My Connections** (or your active environment).
2. Expand **Configuration and Service**.
3. Select **Collection Services**.
4. Right-click any management collection object in the list and select **Properties** to see general information about that collection and a summary of the data that it contains.

You can right-click any collection object and select Create database files to specify the data categories, the range of time within the collection period, and the sampling interval that you want to include in the database files.

You can right-click any collection object and select Graph History to graphically view the data in the management collection object.

> **Related concepts**
>
> "Graph history" on page 96
> Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.
>
> **Related tasks**
>
> "Create database files from Collection Services data" on page 37
> Collection Services places the data you collected into management collection objects.

Delete or keep old management collection objects:

You can delete a collection object from the system by right-clicking the object and selecting **Delete**. If you do not delete the objects manually, Collection Services will delete them automatically after the expiration date and time.

Collection Services deletes only **cycled** management collection objects. A status of **Cycled** means that Collection Services has stopped collecting data and storing it in the object. The status of each management collection object is shown in the list of collection objects when you expand **Configuration and Service** and select **Collection Services**.

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. Even if you move the collection object to another library, Collection Services will delete the object after it expires.

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. Right-click any management collection object in the list and select **Properties** to see the information about that collection. You can specify **Permanent** if you do not want Collection Services to delete your management collection objects for you.

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 fifteen transaction types. Data for additional transaction types is combined and stored as a *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" on page 128
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 API

QYPEENDT, qypeEndTransaction API

QYPELOGT, qypeLogTransaction API

QYPEADDT, qypeAddTracePoint API

Related information

Performance data files: QAPMUSRTNS

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 146.

```
//**********************************************************************
// 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;
```

```c
    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 146.

```
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 */
```

Collecting performance data across partitions

IBM Performance Management for @server iSeries (PM eServer iSeries) automatically triggers
Collection Services to gather nonproprietary performance and capacity data from your server and then
sends the data to IBM for analysis.

One of the analyses PM iSeries provides is to plot the growth of the system to determine when an
upgrade may be necessary. For a system that is not partitioned, this is a straightforward process.
However, if your system has been partitioned into multiple i5/OS partitions, the data arrives at IBM from
each partition separately, which makes forming a reliable view of the entire system performance more
difficult. If the partitions are running AIX® or Linux®, or if any of the i5/OS partitions have PM iSeries
turned off, then no data is sent, which makes forming a view of the entire system nearly impossible.

To address these problems, Collection Services, with IBM Director Multiplatform, can now retrieve data
about CPU usage and number of processors available from your server partitions regardless of the
operating system running on them. PM iSeries summarizes the data before it gets shipped to IBM.
Providing a cross-partition view of CPU utilization helps you and IBM do a much better job of managing
your system resources. This includes balancing workload across the current set of processors as well as
being able to plan for the purchase of more or faster processors when necessary.

How does it work?

The graphic below illustrates how the collection of CPU utilization data across logical partitions works.
The "central system" has the IBM Director Server installed on an i5/OS partition that is running
Collection Services with the *LPAR category selected. Each of the other partitions must have the IBM
Director Agent installed and configured so that IBM Director Server can collect performance data from
them. Each partition must also have the Director Multiplatform extension for Collection Services installed.
IBM Director Server retrieves the CPU utilization data for each partition, including itself, at regular
intervals and stores that data in the Collection Services *MGTCOL object. The data is then processed and
written to the QAPMLPAR database file. Finally, PM iSeries collects and aggregates the data and prepares
to transmit it to IBM. Although this graphic shows Management Central and IBM Electronic Service
Agent™ (ESA) set up to transmit data on the same partition as the IBM Director Server and Collection
Services, the transmission mechanism to IBM could actually be running on a completely different system
and still be set up to gather the cross-partition data from PM iSeries and send it to IBM, business as
usual.
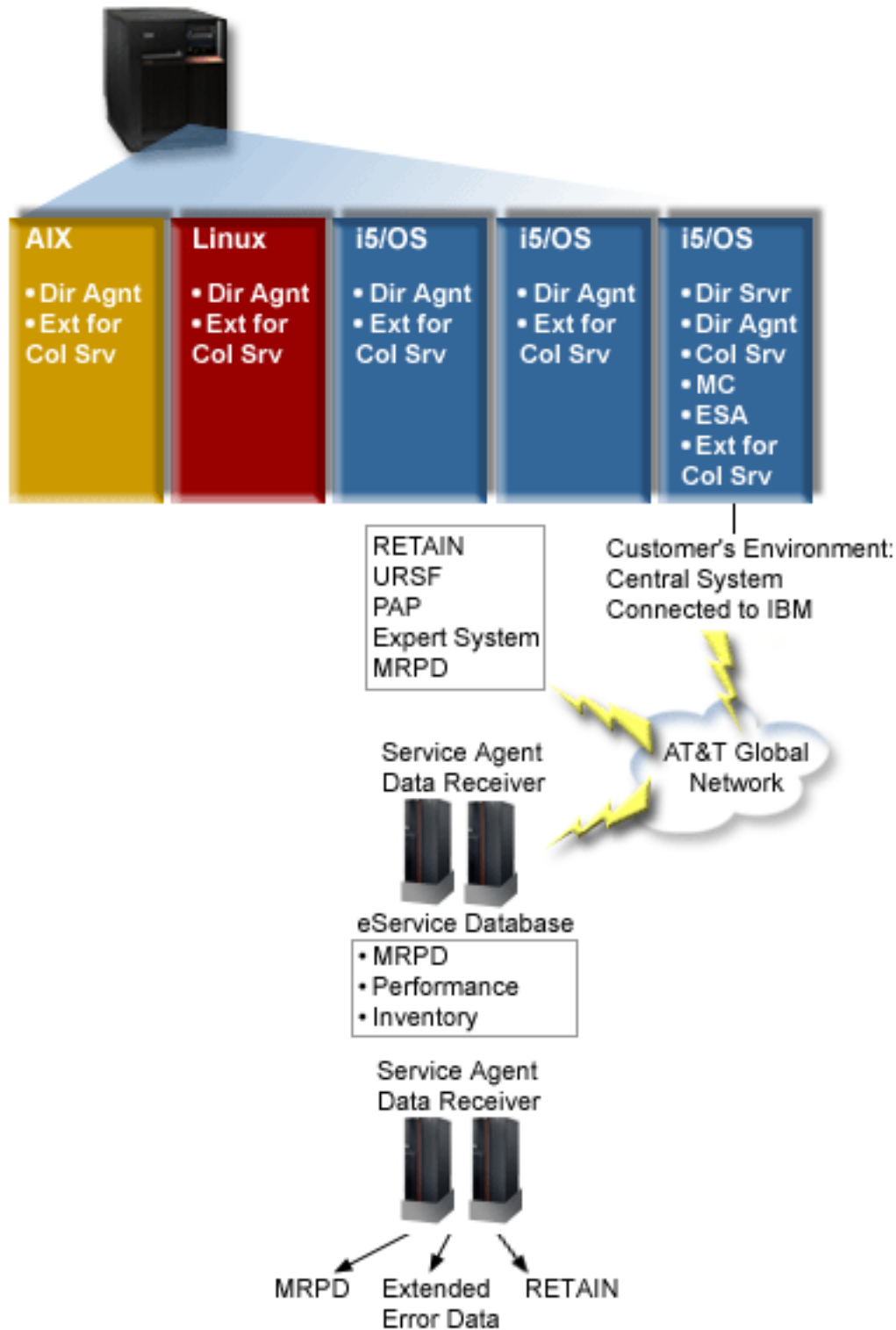
Key
Dir Srvr = IBM Director Server Dir Agnt = IBM Director Agent Col Srv = Collection Services MC = Management Central ESA = IBM Electronic Service Agent Ext for Col Srv = Director Multiplatform extension for Collection Services RETAIN = Remote technical assistance information network URSF = Universal remote support facility MRPD = Machine Reported Product Data

Set it up

The following list provides you with an overview of the steps you must complete to collect performance data across logical partitions:

1. Ensure your IP network is properly configured for all partitions on the same physical system.
2. Ensure you are running a supported operating system on each partition for which you want to collect performance data:
   - i5/OS, Version 5 Release 4
   - AIX 5L™, version 5.3
   - Red Hat Enterprise Linux AS, version 3.0, for IBM PowerPC®
   - SUSE LINUX Enterprise Server 8 for IBM pSeries® and IBM iSeries
   - SUSE LINUX Enterprise Server 9 for IBM pSeries and IBM iSeries
3. Ensure that you have applied the following Collection Services fixes to the partition that acts as your management server:
   - SI12971
   - SI13838 (superseded by SI16328)
   - SI15131 (superseded by SI16499)
   - SI16328 (Linux support)
   - SI16499 (AIX support)

   For the latest information about Collection Services cross-partition support for Linux operating systems, see the informational APAR II13986.

   Go to Fix Central for the latest PTF fixes.
4. Use the Virtualization Engine™ to install IBM Director Server on the i5/OS partition that you want to act as the management server. Consider the management server the central control point that communicates to managed systems, devices, and Collection Services. When the Virtualization Engine installation wizard is complete, IBM Director Server and IBM Director Agent are installed on the i5/OS partition that you want to act as the management server.
5. Install IBM Director Agent on the partitions that you want to be managed by IBM Director Server. These partitions must be on the same physical system as the partition where IBM Director Server is installed.
6. Install IBM Director Console on the system that you want to function as your Director Multiplatform management console.
7. Complete the required configuration steps:
   a. Authorize users for i5/OS on the management partition.
   b. Start Director Multiplatform on each partition.
   c. Start IBM Director console on your management console.

d. In IBM Director Console, add each partition on which you want to monitor performance by right-clicking in the Group Contents pane and selecting **New → IBM Director Systems**.

e. After you have added each partition, request access to manage the partition. In the Group Contents pane right-click the partition and select **Request Access**.

8. On the i5/OS management partition, install the Director Multiplatform extension for Collection Services by copying the necessary files for Collection Services from the Collection Services directory to the appropriate Director Multiplatform directory. The Collection Service files are ColSrvLparDataExt.TWGExt, ColSrvLparDataSubagt.TWGSubagent, and ColSrvDir.jar. Copy the Collection Services files using the following commands:

```
CPY
OBJ('/qibm/proddata/os400/collectionservices/lib/ColSrvLparDataExt.TWGExt')
    TODIR('/qibm/userdata/director/classes/extensions')

CPY
OBJ('/qibm/proddata/os400/collectionservices/lib/ColSrvLparDataSubagt.TWGSubagent')
    TODIR('/qibm/userdata/director/classes/extensions')

CPY
OBJ('/qibm/proddata/os400/collectionservices/lib/ColSrvDir.jar')
    TODIR('/qibm/userdata/director/classes')
```

9. Distribute the Collection Services files from the management partition to the i5/OS partitions from which you plan to collect performance data. You can do this by File Transfer Protocol (FTP) with the binary option, or by mapping a drive and copying the files to the file system, or by any other distribution mechanism that you might have in place. You can access the files on the i5/OS management partition in the directory, /qibm/proddata/os400/collectionservices/lib.

a. Distribute ColSrvLparDataExt.TWGExt to the Director Multiplatform extensions directory, /qibm/userdata/director/classes/extensions, on the i5/OS partition that you want to manage.

b. Distribute ColSrvLparDataSubagt.TWGSubagent to the Director Multiplatform extensions directory, /qibm/userdata/director/classes/extensions, on the i5/OS partition that you want to manage.

c. Distribute ColSrvDir.jar to Director Multiplatform classes directory, /qibm/userdata/director/classes, on the i5/OS partition that you want to manage.

10. On each Linux partition, install the Director Multiplatform extension for Collection Services by installing the Collection Services RPM file, ColSrvDirExt.rpm.

a. Distribute the Collection Services RPM file from the management partition to the Linux partitions from which you plan to collect performance data. You can do this by File Transfer Protocol (FTP) with the binary option, or by mapping a drive and copying the files to the file system, or by any other distribution mechanism that you might have in place. You can use Qshell to access the RPM file in the i5/OS management partition directory, /qibm/proddata/os400/collectionservices/lib/ColSrvDirExt.rpm.

b. On each Linux partition, run the following command from the directory where the RPM file exists:

```
rpm -Uhv --force ColSrvDirExt.rpm
```

11. On each AIX partition, install the Director Multiplatform extension for Collection Services by installing the Collection Services package, aix-ColSrvDirExt.bff.

a. Distribute the Collection Services package file from the management server to the AIX partitions from which you plan to collect performance data. You can do this by File Transfer Protocol (FTP) with the binary option, or by mapping a drive and copying the files to the file system, or by any other distribution mechanism that you might have in place. You can use Qshell to access the package file in the i5/OS management partition directory, /qibm/proddata/os400/collectionservices/lib/aix-ColSrvDirExt.bff.

b. On each AIX partition, run the following command from the directory where the BFF file exists:

```
installp -Fac -d  aix-ColSrvDirExt.bff ColSrvDirExt
```

12. In IBM Director Console, update the collection inventory on each partition by right-clicking the partition and selecting **Perform Inventory Collection**.

13. Activate PM iSeries, which automates the start of Collection Services and then creates the database files during collection. If PM iSeries is already running, use the following Start Performance Collection (STRPFRCOL) command:

```
STRPFRCOL CYCCOL(*YES)
```

You can use the following tools to track performance data across partitions:

- Collection Services
- PM iSeries
- IBM eServer Workload Estimator

Related concepts

Configured

"IBM Performance Management for eServer iSeries" on page 99
PM iSeries automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity. PM iSeries uses the performance data collected by Collection Services.

Management Central

Related tasks

"Activate PM iSeries" on page 102
PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

Partitioning the server

"Send PM iSeries data with Service Agent over Extreme Support (Universal Connection)" on page 104
PM iSeries uses Collection Services to gather the nonproprietary performance and capacity data from your server. After you have collected this data, you can use Electronic Service Agent over Extreme Support to send the data to IBM.

Related information

Install IBM Director Server

Install IBM Director Agent

Install IBM Director Console

Authorize users for OS/400

Start Director Multiplatform

Start IBM Director console

IBM Director Multiplatform

IBM Virtualization Engine

Domino 6 for iSeries Best Practices Guide (Workload Estimator)

Find 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.

Another tool that shows job wait statistics is Job Watcher. Job Watcher is a component of the iDoctor for iSeries software product. Job Watcher returns realtime information about a selected set of jobs, threads and licensed internal program (LIC) tasks. At specified time intervals, Job Watcher samples anywhere from one thread/job to all threads/job. Job Watcher gathers a variety of performance data, including detailed wait statistics for jobs, tasks, and threads.

Two types of wait buckets accumulate wait state data:

| **Static wait buckets**
| A set of 16 statistical buckets which accumulate wait state data. Static wait buckets, used by
| Collection Services, provide a stable view of the wait state data. Data from the static buckets is
| reported in the QAPMJOBWT file.

| **Dynamic wait buckets**
| A set of up to 32 statistical buckets which accumulate wait state data. Initially, dynamic wait
| buckets are mapped to contain the same data as the static wait buckets. However, you can remap
| dynamic wait buckets.

| **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 65 535-binary data.

> **Related concepts**
>
> "iDoctor for iSeries" on page 137
> The iDoctor for iSeries plugin consists of a variety of software tools for managing performance, for
> example, PEX Analyzer for detailed trace data analysis and Job Watcher for trace-level information
> about a job's behavior.
>
> **Related information**
>
> Performance data files: QAPMJOBWT
>
> Performance data files: QAPMJOBWTD
>
> A jobs life
>
> Work management job attributes

Understanding disk consumption by Collection Services

The amount of disk resource Collection Services consumes 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 data collection. Next, establish a base size for one
day's worth of data collection 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 a *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

Interval rate	Intervals per collection	Multiplier	Size in MB
1 minutes	1440	15	7500
30 seconds	2880	30	15000
15 Seconds	5760	60	30000

The size of a *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, a *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 will cause excess data consumption.

Collect 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.

Enterprise Workload Manager (EWLM) is a robust performance management tool that allows you to view the performance of work that runs in your business environment. Furthermore, EWLM allows you to assign performance goals to specific work. This allows you to monitor application-level transactions

entity. EWLM encourages middleware and third-party software vendors to instrument their applications
with the Open Group Application Response Measurement 4.0 (ARM) APIs. EWLM uses the ARM APIs to
collect detailed performance data from ARM applications.

To learn more about ARM APIs, visit The Open Group Web site at www.theopengroup.org/arm.

Related information

Performance data files: QAPMARMTRT

Performance data files: QAPMUSRTNS

Enterprise Workload Manager

Intelligent Agents

The Intelligent Agents console for iSeries Navigator provides system administrators with an easy way to manage one or more ABLE (Agent Building and Learning Environment) agents running on a single system or across different systems.

Intelligent agents are Java-based software components that are capable of learning certain behaviors over time through complex autonomic algorithms. Intelligent agents can have many different capabilities, from simply monitoring for certain events to more complex actions like analyzing network problems, preventing unplanned system restarts, or managing storage. Although the goal of using agents is to simplify the system administrators tasks through autonomic computing, system administrators still need a way of starting, stopping, responding to, and monitoring the actions of their agents.

The Intelligent Agents console for iSeries Navigator provides system administrators with an easy way to manage one or more ABLE (Agent Building and Learning Environment) agents running on a single system or across different systems. After the agent console connects to the agent services that exist across your domain, you can monitor and work with any number of pre-configured agents on any of the systems in your domain.

Intelligent Agent concepts

The Intelligent Agents console uses ABLE agents running on or across a distributed agent platform. Find out more about ABLE agents, and the agent services that make up the distributed platform.

ABLE agents:

The Intelligent Agents console for iSeries Navigator works with Agent Building and Learning Environment (ABLE) agents.

ABLE agents are Java objects capable of automating tasks through the use of rule-based reasoning and learning certain behaviors over time by using data mining algorithms contained in the ABLE component library. ABLE is a Java framework and toolkit used for building multiagent intelligent autonomic systems, and provides specific support for developing agents that work with the iSeries Navigator Intelligent Agent platform and console. Intelligent agents developed using ABLE can have the following capabilities:

- Learn from experience and predict future states
- Analyze metric data using classification and clustering algorithms to detect complex states and diagnose problems
- Interface with other autonomic components via web services
- Reason using domain-specific Java application objects
- Use powerful machine reasoning, including: Boolean forward and backward chaining, predicate logic (Prolog), Rete'-based pattern match, and fuzzy systems
- Have autonomous (proactive) behavior and goals
- Correlate events into situations, reason, and take actions

The ABLE toolkit contains several examples of how to design your own agent, and an iSeries template agent is included that you can use as a model when developing your own agent. To create an agent that can be fully managed from the console, the agent should extend **AbleEServerDefaultAgent**.

Related concepts

"Develop Agents" on page 69
Create and customize your own agent to perform the tasks that you desire. The ABLE toolkit and its associated documentation provide a working development environment and a template agent that can be used as a guide for developing your own agents.

Agent platform:

Agent Services live on your system or across your distributed platform, and are responsible for the life cycle, security, and behavior of your agent.

The Intelligent Agents console in iSeries Navigator requires an agent platform be configured on your system, or across a distributed network. An agent platform is nothing more than a set of Java Virtual Machines, or agent pools, that run the services and agents of the platform. The platform is defined by a preferences file called **ableplatform.preferences**. This file lists the location (system and port) of each agent pool (JVM), the services that will run on or across the platform, and the agents that are allowed to run in the platform. If security is configured, the preferences file also lists the Kerberos user and service principals used to authenticate each service, agent, and user that is part of the platform.

Agent services, which can exist on any of the systems across your distributed platform, are responsible for the life cycle, security, and behavior of your agent. Agents running on the same system or distributed agents running across different systems use the defined set of platform services for different tasks such as getting a unique name, looking up other agents in a directory, logging, and passing messages to another agent.
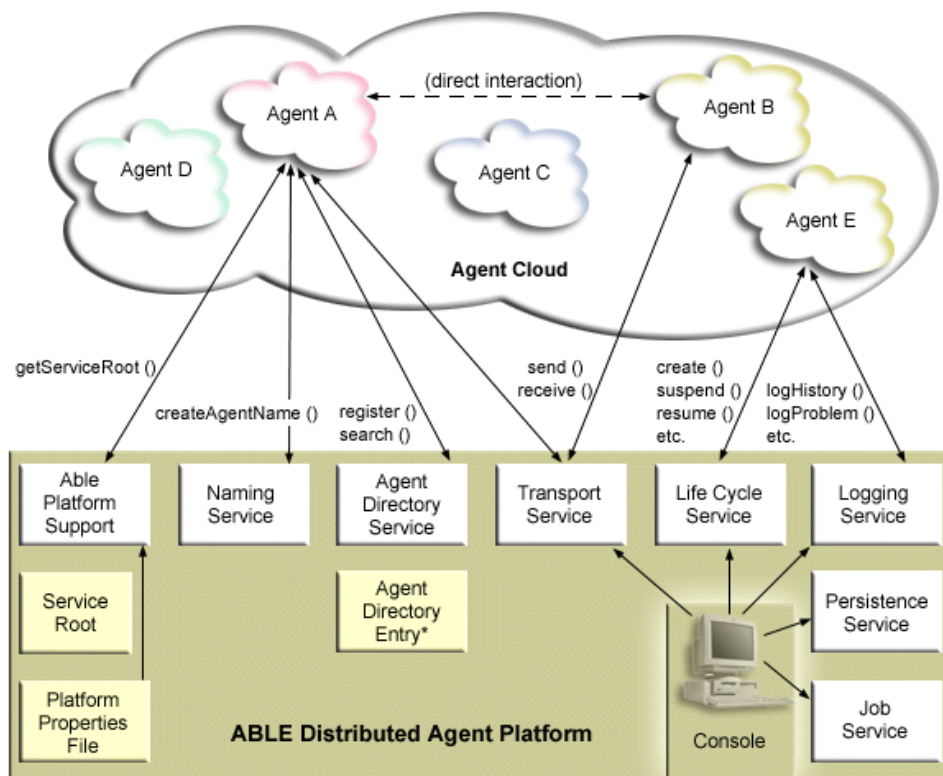
The following services are made available to the agents running on or across a platform and to the users connected to the platform:

- **Naming Service**

This service provides the creation of a globally unique name among all other pieces in the distributed platform. The Naming service also provides security for the platform when security is turned on. Kerberos is used when starting the platform to authenticate all services, pools, and users. Throughout the life of the platform, this service will also act as the trusted third party to secure all interactions between the platform's agents, services, and users.

- **Directory Service**

  When an agent wants to make itself known to other services and agents across the platform, it creates an agent description and registers this description to the directory service. After the agent is registered, descriptions can be modified and removed.

- **Lifecycle Service**

  This service is used to manage agents. Agents can be created, started, suspended, resumed and destroyed through this service.

- **Transport Service**

  This service provides locators for parts of the platform. Inter-agent communication is also made available by this service.

- **Logging Service**

  A running agent may encounter a problem that requires outside intervention. The Logging Service creates and logs requests, and handles the corresponding answers that are sent back to it from the request. The progress of an agent can also be logged to this service for others to view.

- **eServer Job Service**

  The different services and jobs of the platform register their job entry to this service. This service provides critical information about the platform when the platform is running on an iSeries.

- **Persistence Service**

  Services and agents may use this service to persist valuable information. The Naming, Directory, Lifecycle, Logging and Job Services can be backed up and stored in a database when the Persistence Service is configured.

  Related tasks

  "Configure your agent platform" on page 72
  Provides a brief overview of the agent platform, and then provides detailed configuration steps for modifying the platform preferences file. Before you begin using the Intelligent Agents console in iSeries Navigator, you first need to configure the agent platform.

  Related reference

  "Set up your agent environment" on page 71
  Before you can begin managing your agents with the Intelligent Agents console, you will need to configure your agents and agent services (the agent platform) to run on or across the systems in your environment. A secure environment requires Kerberos and additional platform configuration.

Develop Agents

Create and customize your own agent to perform the tasks that you desire. The ABLE toolkit and its associated documentation provide a working development environment and a template agent that can be used as a guide for developing your own agents.

You can use the Agent Building and Learning Environment (ABLE) toolkit to develop your own hybrid intelligent agents. This Java framework has its own rule language (ARL) and its own GUI-based interactive development environment, the Able Agent Editor; both are provided to assist in the construction of Able agents.

ABLE 2.0

Both the ABLE toolkit and complete ABLE documentation are available to download in .zip packages.

The iSeries Navigator Intelligent Agents console ships with a template agent that you can use as a guideline for developing agents to work with the console. The source code for **AbleEserverTemplateAgent** is stored in **ableplatform.jar**, located in **QIBM/ProdData/OS400/Able**.

AbleEserverTemplateAgent makes use of many of the features available when developing agents using the ABLE framework. It demonstrates how an agent would create a set of capabilities that could be managed through the console. It includes a **Customize** panel that can be used to alter agent settings and an **About** panel that is used to display information about the agent. It also shows how an agent uses the Logging Service to log requests and history entries that can be displayed and responded to through the console.

Agent Capabilities

The EServerTemplateAgent has the following capabilities:

- **Time Monitor**

  The agent will watch for minute and hour changing events and take action. There are four different situations that the agent will follow depending on what the capability is set to, or how the user responds to a request if one is logged:

  1. Log the change without telling the time.
  2. Log the change with telling the time as a long.
  3. Log the change with telling the time in MM/DD/YY format
  4. Do nothing

- **Duplicate Request**

  The agent will watch for multiple hour and minute change requests. There are two different situation that the agent will follow with this capability if a duplicate is found.

  1. Create a duplicate request
  2. Do not create a duplicate request

Customization Panel

The agent supplies a customization panel that allows you to adjust the interval at which the agent will check if the minute or hour has changed.
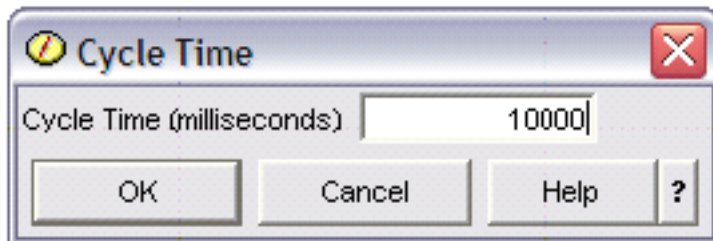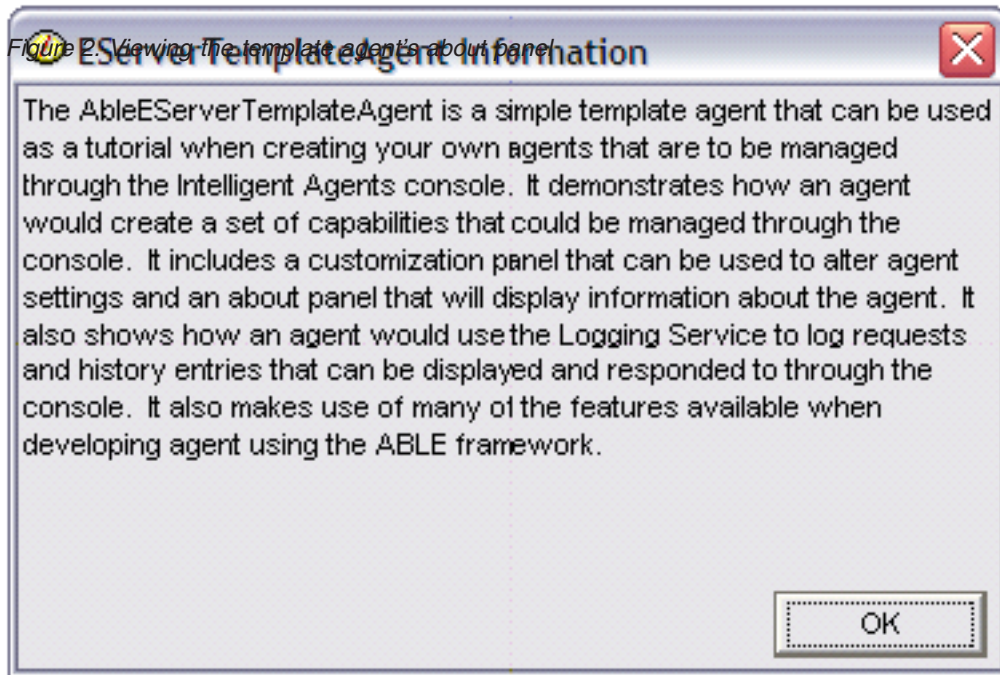

Figure 1. An example use of the Customization Panel

About Panel

The agent supplies an about panel that allows you to provide detailed information about the agent.

Figure 25. Viewing the template agent's about panel

EServer Template Agent Information

The AbleEServerTemplateAgent is a simple template agent that can be used as a tutorial when creating your own agents that are to be managed through the Intelligent Agents console. It demonstrates how an agent would create a set of capabilities that could be managed through the console. It includes a customization panel that can be used to alter agent settings and an about panel that will display information about the agent. It also shows how an agent would use the Logging Service to log requests and history entries that can be displayed and responded to through the console. It also makes use of many of the features available when developing agent using the ABLE framework.

OK

Related concepts

"ABLE agents" on page 67
The Intelligent Agents console for iSeries Navigator works with Agent Building and Learning Environment (ABLE) agents.

Agent Learning and Building Environment 2.0:

Agent Learning and Building Environment (ABLE) is a Java framework, component library, and productivity tool kit for building intelligent agents using machine learning and reasoning.

Both the ABLE 2.0 Toolkit and the ABLE Documentation bundle are available to download as .zip packages:

* ABLE 2.0 Toolkit: AbleAll_2.0.0.zip

    This 6 MB zipped package contains the ABLE Java framework, component library, and tool kit.

* ABLE Documentation: doc.zip

    This 12 MB zipped package contains complete ABLE documentation, including an FAQ, the README, license agreement, JavaDoc, and more. Also included in doc.zip is a second zipped package (Able-Class.zip) that contains several excercises and presentations designed to help you develop ABLE agents.

Set up your agent environment

Before you can begin managing your agents with the Intelligent Agents console, you will need to configure your agents and agent services (the agent platform) to run on or across the systems in your environment. A secure environment requires Kerberos and additional platform configuration.

The iSeries Navigator Intelligent Agents console functions by connecting to an agent platform running on your system, or across a distributed network. The agent platform defines the agent pools (JVMs) that your agent services and agents will run in. Before you begin setting up your agent platform, you will

need to determine your security preferences. A secure platform requires that you configure Kerberos. The following topics provide detailed steps for setting up your agent platform and configuring security:

Related concepts

"Agent platform" on page 68
Agent Services live on your system or across your distributed platform, and are responsible for the life cycle, security, and behavior of your agent.

Configure your agent platform:

Provides a brief overview of the agent platform, and then provides detailed configuration steps for modifying the platform preferences file. Before you begin using the Intelligent Agents console in iSeries Navigator, you first need to configure the agent platform.

Agent platform overview

To manage agents using the intelligent agents console, you must first define, secure, and start an agent platform that the console will connect to. An agent platform is nothing more than a set of Java Virtual Machines, or agent pools, that run the services and agents of the platform. The **ableplatform.preferences** and **able.preferences** files are used to define a platform.

In its simplest form, with security turned off, **ableplatform.preferences** defines:
- The location (system and port) of each Pool.
- The services that will run in the platform.
- The agents that are allowed to run in the platform.

Once the agent platform is set up, the services that run on or across the platform allow an agent to receive a unique name, look up other agents in a directory, log history or requests, pass messages to one another, or control the state of an agent. For a conceptual overview of the distributed platform and more information about the available agent services, see the following concept article: Agent platform.

Define the agent platform

To begin configuring your platform, you must define your agent pools, agent services, permitted agents, and add Kerberos security principals by modifying the following file: **ableplatform.preferences**.

The default location of **ableplatform.preferences** is **QIBM/ProdData/OS400/Able**.

Note: Multiple platforms can be configured, and you need to ensure that your platform does not reside at the same location as an existing platform using the same port. See the Start the agent platform topic for more details.

The following code samples taken from **ableplatform.preferences** provide examples of how to modify the platform preferences:

Note: When you open the file and begin making changes to the content, understand that small errors and misspellings will cause the agent platform to fail, and there is currently no easy way to debug your mistakes. Avoid commenting out properties that are unused, commenting out an unused property can cause the platform to fail. For example, if you choose to run the platform with security turned off, do not comment out the principal properties through the file.

1. **Define agent pools**

   A platform is nothing more than a set of distributed Java Virtual Machines. Each JVM is called an agent pool, and each JVM or pool can host multiple services and agents (an agent pool does not have to host service, it could be used to run just agents). You must define the location of each of your Java Virtual Machines (agent pools) in the preferences file by specifying the IP address (fully qualified

system name) and port. Also, specify an Alias (any unique name) for each agent pool. When security is turned on, you must associate a service principal with each agent pool; for more information about using Kerberos service principals, see the Secure your agent environment topic. The following is an example of how a set of agent pools could be defined:

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

```
#----------------------------------------------------------------------
# Java Virtual Machines
#----------------------------------------------------------------------
AgentPool.1.Alias     = Pool1
AgentPool.1.IpAddress = systemname.ibm.com
AgentPool.1.Port      = 55551
AgentPool.1.Principal = servicePrincipal1

AgentPool.2.Alias     = Pool2
AgentPool.2.IpAddress = systemname.ibm.com
AgentPool.2.Port      = 55552
AgentPool.2.Principal = servicePrincipal1

AgentPool.3.Alias     = Pool3
AgentPool.3.IpAddress = systemname.ibm.com
AgentPool.3.Port      = 55553
AgentPool.3.Principal = servicePrincipal2
#----------------------------------------------------------------------
```

2. **Define agent services**

   Define the agent services that you want to run on the platform, and specify the alias of the agent pool you want them to run in. Each agent service must point to a factory; the factory is a Java Class that creates the agent service. The Persistence service is used to restart a platform to its previous state. Specify to turn persistence on or off. If you turn persistence on, you must specify a Database, Table and Schema so that persistence has a location to store backed up data on. You can also specify a value for the PersistenceRetry property. If the persistence service fails and you specified a value of 5000 for the PersistenceRetry property, it will attempt to retry every 5000 milliseconds. The following code example shows how three different services, Directory, Logging, and Persistence could be defined:

```
Services=Agent-Directory-Service,Agent-Logging-Service,
Persistence-Service

Agent-Directory-Service.AgentPool        = Pool1
Agent-Directory-Service.Factory =
com.ibm.able.platform.RMIVerifiableDirectoryServiceFactory
Agent-Directory-Service.Persistence      = off
Agent-Directory-Service.PersistenceDatabase = *LOCAL
Agent-Directory-Service.PersistenceTable  = qahadir
Agent-Directory-Service.PersistenceSchema = QUSRSYS
Agent-Directory-Service.PersistenceRetry  = 5000

Agent-Logging-Service.AgentPool          = Pool1
Agent-Logging-Service.Factory =
com.ibm.able.platform.RmiAgentLoggingServiceFactory
Agent-Logging-Service.Persistence        = off
Agent-Logging-Service.PersistenceDatabase = *LOCAL
Agent-Logging-Service.PersistenceTable    = qahalog
Agent-Logging-Service.PersistenceSchema   = QUSRSYS
Agent-Logging-Service.PersistenceRetry    = 5000
Agent-Logging-Service.Properties          = history-log-max : 100
```

Note: You can specify to control performance by adding a history-log-max property to the Logging service. If you specify history-log-max=100, each agent will keep only its 100 most current history logs.

```
Persistence-Service.AgentPool    = Pool1
Persistence-Service.Factory =
com.ibm.able.platform.RmiPlatformPersistenceServiceFactory
Persistence-Service.Properties   =
persistence-driver : com.ibm.db2.jdbc.app.DB2Driver,
persistence-protocol : jdbc,
persistence-subProtocol : db2,
blob-type : BLOB,
persistence-dbFlushTime : 1000,
persistence-dbResetAll : off
```

The Persistence service provides backup and recovery for your agent platform. To use persistence with agent services running on or across your platform, you need to define several Persistence-Service.Properties:

- **persistence-driver**

  Defines the JDBC driver that the persistence service will use. By default the persistence-driver is set to use the native DB2 driver.

- **persistence-protocol and subProtocol**

  Defines the database protocol that the persistence service will use. By default the protocol is set to jdbc and the subProtocol is set to db2.

- **blob-type**

  Defines the blob type associated with the JDBC driver you are using. The default for DB2 is set to BLOB, but if you choose to use a different database like CloudScape for example, you would define blob type as blob-type : LONG VARBINARY.

- **persistence-dbFlushTime**

  Specifies the rate at which you want the persistence service to flush data to the database in milliseconds.

- **persistence-dbResetAll**

  If you specify to turn this property **on**, when you restart the platform all previously persisted data will be cleared from the database.

3. **Define permitted agents**

   You must define all of the agents that you want to allow access to the platform and the agent services running on or across the platform. The following is an example of how an agent could be defined. More details about each agent property are listed after the following example:

```
Agent.1.Alias=Agent1
Agent.1.AutonomyLevel=Medium
Agent.1.ClassName=
com.ibm.able.platform.examples.EServerTemplateAgent
Agent.1.ConstructorArgs=String:agentName
Agent.1.EligiblePrincipals=principalAlias1, principalAlias2
Agent.1.EligibleAgentPools=pool1, pool2, pool3
Agent.1.InitArgs=
Agent.1.LastChangedDate=January 11, 2003 11:11am
Agent.1.Type=Tester1
Agent.1.Vendor=IBM1
Agent.1.Version=1.1
```

- **Alias**

  Provide a unique name for your agent. This name will be used by the agent console.

- **AutonomyLevel**

  Specify the agents initial autonomy level. A user can change this setting from the console. Determine the level of independence you want to associate with your agent, and set the automation level accordingly. The higher you set the automation level, the less your agent will request permission to take an action. If you set an agent to **High automation**, it will perform most actions without requesting a response first. If you are concerned about an agent's behavior, you may want to lower the automation level, (increasing the frequency by which the agent requests permission to take action), by changing the setting to **Medium automation**.

- **ClassName**

  Specifies the the actual agent Java Class.
- **ConstructorArgs**

  Allows you to provide arguments in the properties file that you want to pass to your agent.
- **EligiblePrincipals**

  When security is turned on, you must define who has authority to start an instance of your agent by associating one or more user principal aliases with each agent; for more information about using Kerberos service principals, see the Secure your agent environment topic.
- **EligibleAgentPools**

  Specify the alias of one or more agent pools that you want to use to run your agents on the platform.
- **InitArgs**

  Allows you to pass in any Init arguments to your agent from the preferences file.
4. **Secure your agent platform**

   After you have defined your agent pools, agent services, and permitted agents, you may want to configure security on the platform. For more information on Kerberos principals, trustlevels, and how they are used and defined to secure the agent platform, see: Secure your agent environment.

After you have defined your agent pools, agent services, and permitted agents, and optionally set up security, you need to Start the agent platform.

Related concepts

"Agent platform" on page 68
Agent Services live on your system or across your distributed platform, and are responsible for the life cycle, security, and behavior of your agent.

"Start the agent platform" on page 80
After you define the agent platform and optionally secure your platform, you will need to start all the Java Virtual Machines associated with your agent services using iSeries CL commands.

Related tasks

"Secure your agent environment"
It is strongly recommended that you use Kerberos user and service principals to authenticate users, agent pools, and agent services to one another on or across a secure platform or distributed platform.

Secure your agent environment:

It is strongly recommended that you use Kerberos user and service principals to authenticate users, agent pools, and agent services to one another on or across a secure platform or distributed platform.

Platform security can be turned on or off. If you choose to run on or across a platform that has security turned off, anyone can deregister or modify another person's agent descriptions. Anyone can change the capabilities or state of any agent. Anyone can remove or answer any requests, even if they are not their own. Agents can potentially take destructive actions when being used incorrectly or by the wrong user. To ensure that agents are used the way they were intended, security features have been added to the infrastructure of the platform.

When security is turned on, agents and services will be able to authenticate and authorize every action that is taken on or across the platform. An agent can only deregister or alter its own agent description, an agent must authorize all answered requests and capability changes, and a certain authority level will be required to alter the state of an agent. The use of an agent can be limited to certain users and locations. When security is turned on, every action that occurs can be traced back to a known user so platform authentication and authorization can occur.

If you choose to secure your agent platform, you can turn security on by changing the Security property to **Security=on** in the **able.preferences** file that defines your platform.

Related concepts

"Start the agent platform" on page 80
After you define the agent platform and optionally secure your platform, you will need to start all the Java Virtual Machines associated with your agent services using iSeries CL commands.

Related tasks

"Configure your agent platform" on page 72
Provides a brief overview of the agent platform, and then provides detailed configuration steps for modifying the platform preferences file. Before you begin using the Intelligent Agents console in iSeries Navigator, you first need to configure the agent platform.

Configure your platform to use Kerberos:

The intelligent agent platform uses Kerberos principals to authenticate users and services throughout the agent platform. Kerberos protocol, developed by Massachusetts Institute of Technology, allows a principal (a user or service) to prove its identity to another service within an insecure network.

Authentication of principals is completed through a centralized server called a key distribution center (KDC). The KDC authenticates a user with a Kerberos ticket. These tickets prove the principal's identity to other services in a network. After a principal is authenticated by these tickets, they can exchange encrypted data with a target service.

The platform uses Kerberos to authenticate user signon and initial platform startup. To use Kerberos to secure your platform, you must either find an existing KDC, or create a working KDC that all parts of the platform will use. Every system running a piece of the platform and every PC running a console that connects to this platform must be configured to use this KDC. You need to list all Kerberos principals in the **ableplatform.preferences** file that are used by the platform to authenticate users and services. Each platform Java Virtual Machine (agent pool) will have a service principal associated with it, and each user logging onto the platform from a console will need a user principal. All of these principals will need to be added to the KDC.

1. **Find or create a usable Kerberos key distribution center (KDC)**

   The agent platform does not require a KDC on i5/OS, a KDC running on any platform will work. If you cannot find an existing KDC to use, you can create your own. In V5R3 or later, i5/OS supports a native Kerberos server in i5/OS PASE. You can configure and manage a Kerberos server from your iSeries system. To configure a Kerberos server in i5/OS PASE, complete the following tasks:

   a. In a character-based interface, type: **call QP2TERM**. This command opens an interactive shell environment that allows you to work with i5/OS PASE applications.

   b. At the command line, enter: **export PATH=$PATH:/usr/krb5/sbin**. This command points to the Kerberos scripts that are necessary to run the executable files.

   c. At the command line, enter: **config.krb5 -S -d iseriesa.myco.com -r MYCO.COM**. This command updates the krb5.config file with the domain name and realm for the Kerberos server, creates the Kerberos database within the integrated file system, and configures the Kerberos server in i5/OS PASE. You will be prompted to add a database Master Password and a password for the admin/admin principal which is used to administer the Kerberos server.

   d. At the command line, enter: **/usr/krb5/sbin/start.krb5** to start the servers.

2. **Configure systems in your agent environment to use Kerberos**

   After you create a Kerberos server (KDC), you need to individually configure each client PC that will attempt to connect to the secure platform, and each iSeries system in your agent platform to point to your Kerberos server (KDC).

   • **Configure your client PC**

To configure a client PC, you need to create a text file called **krb5.conf** in the security folder of the JVM that runs your iSeries Navigator intelligent agents console located here (where C: is the drive your Client Access driver is installed on):

C:\Program Files\IBM\Client Access\JRE\Lib\Security

The **krb5.conf** file tells all JVMs started from this JRE which KDC to use when dealing with Kerberos. The following is an example of what a generic **krb5.conf** file might look like if the KDC realm was KDC_REALM.PASE.COM and was found on system1.ibm.com:

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

```
[libdefaults]
 default_realm          = KDC_REALM.PASE.COM
 default_tkt_enctypes   = des-cbc-crc
 default_tgs_enctypes   = des-cbc-crc

[realms]
 KDC_REALM.PASE.COM = {
        kdc = system1.rchland.ibm.com:88
}

[domain_realm]
 .rchland.ibm.com = KDC_REALM.PASE.COM
```

- **Configure your iSeries system**

  To point your iSeries system to your KDC, you need to modify the following file:

  /QIBM/userdata/OS400/networkauthentication/ **krb5.conf**

  The **krb5.conf** file tells all JVMs started from this JRE which KDC to use when dealing with Kerberos. The following is an example of what a generic **krb5.conf** file might look like on the server if the KDC realm was KDC_REALM.PASE.COM and was found on system1.ibm.com:

```
??(libdefaults??)
  default_realm = KDC_REALM.PASE.COM
??(appdefaults??)
??(realms??)
  KDC_REALM.PASE.COM = {
    kdc = system1.rchland.ibm.com:88
  }
??(domain_realm??)
  system1.rchland.ibm.com = KDC_REALM.PASE.COM
```

3. **Acquire Kerberos user and service principals**

   After you configure a KDC, you will need to create the user and service principals you plan to use to secure the platform, and register these principals to the KDC:

   Service Principals:
   > Each agent pool (JVM) defined in **ableplatform.preferences** must have a service principal associated with it. Service principals are specific to the system that they will run on, so they must include that system name and be in the following format: **ServicePrincipalName/systemName@KDCRealm**. Each of the agent pools on the platform can use the same service principal, or you can specify that each pool use its own service principal. If each of your agent pools have different authority levels, then different principals should be used for each different authority level.

   User Principals:
   > Each user that you want to allow to connect to the secure platform through the console will need a user principal. User principals can be associated with each agent definition listed in **ableplatform.preferences**. A user principal can connect to a platform from the console,

regardless of the system the console is running on. Because of this, a user principal only needs to include the principal name and the KDC realm the principal belongs to: **UserPrincipalName@KDCRealm**.

You need to add a principal to the KDC for each Service and User principal that your platform will use. The following steps will help you add your principals to your KDC if you are using the native KDC on the server:

a. In a character-based interface, type: `call QP2TERM`.

b. At the command line, enter: `export PATH=$PATH:/usr/krb5/sbin`. This command points to the Kerberos scripts that are necessary to run the executable files.

c. At the command line, type: `kadmin -p admin/admin`, and press **Enter**.

d. Sign in with administrator's password.

e. At the command line:

   • To add service principals for Pools running on an iSeries server:

   `addprinc -pw secret servicePrincipalName/iSeries fully qualified host name@REALM`

   • To add user principals:

   `addprinc -pw secret jonesm`. This creates a principal for a user to log in from a console.

   • To add service principals for Pools running on a PC:

   `addprinc -requires_preauth -e des-cbc-crc:normal -pw host/pc1.myco.com.`

If you are using the native KDC, see the following topics for more information on how to add principals to your KDC:

If you are adding Service principals for Pools that will be running on an iSeries server, see: Add i5/OS principals to the Kerberos server.

If you are adding User principals or Service principals for Pools that will be running on a PC, see: Create Host principals for Windows® 2000 workstations and users

4. **Add service principals to each keytab file**

   When starting up a secure platform each agent pool will use the principal that it was defined to start with, and use it to authenticate itself. This requires each Pool JVM to have access to valid Kerberos credentials for the principal it is using. The iSeries Start Agent Services (STRAGTSRV) command will handle this, as long as there is an entry in the keytab file for the principal that is being used. Follow these steps to add an entry to the keytab file for each service principal that is to run on each of your platform systems:

   If you are running the native KDC on an iSeries server:

   a. In a character-based interface, type: `STRQSH`. This command starts the qsh shell interpreter.

   b. Enter the following command (where *ServicePrincipal* is the name of the service principal you want to add, *system@KDCRealm* is the fully qualified system name and Kerberos realm, and where *thePassword* is the password associated with your service principal): `keytab add ServicePrincipal/system@KDCRealm -p thePassword`

After you set up your KDC and create your user and service principals, you need to configure security in your **ableplatform.preferences** file.

   Related tasks

   Configure a Kerberos server in i5/OS PASE

   Configure network authentication

   "Configure platform security"
   Before you begin, ensure that you have configured your Kerberos key distribution center (KDC).

Configure platform security:

Before you begin, ensure that you have configured your Kerberos key distribution center (KDC).

When security is turned on, **ableplatform.preferences** acts as a policy file for the security of the platform it defines. The following steps provide examples for how principals, trust levels, and permissions could be configured:

1. **Define User and Service principals**

   After you acquire user and service principals, and register them with your KDC, you need to add these principals to **ableplatform.preferences** . When security is turned on, a user must be defined with a valid Kerberos user principal to gain access to the platform, and all agent services and agent pools must have a valid Kerberos service principal assigned to them. Add the user or service principals you have registered with your KDC, and specify an alias for each principal (the alias can be any unique name you want to use):

   Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

   ```
   #----------------------------------------------------------------------
   # Principals
   #----------------------------------------------------------------------
   Principal.1.Alias     = servicePrincipal1
   Principal.1.Principal = name1/systemName@REALM

   Principal.2.Alias     = servicePrincipal2
   Principal.2.Principal = name2/systemName@REALM

   Principal.3.Alias     = userPrincipal1
   Principal.3.Principal = name1@REALM

   Principal.4.Alias     = userPrincipal2
   Principal.4.Principal = name2@REALM
   ```

2. **Define trust levels**

   After you add user and service principals, you need to define the trust level associated with each principal. A trust level is associated with a principal to help define the capabilities of a user or service on the platform. Associating a trust level with a principal is also a way to group principals. The same trust level can be associate with multiple user and service principals. Add the principal alias you assigned to your service and user principals in step 1, (comma delineated), to the trust level you want to associate it with, and provide a unique name for trust level alias:

   ```
   #----------------------------------------------------------------------
   # Trust Levels
   #----------------------------------------------------------------------
   TrustLevel.1.Alias      = HighlyTrusted
   TrustLevel.1.Principals = servicePrincipal1,userPrincipal1

   TrustLevel.2.Alias      = SomewhatTrusted
   TrustLevel.2.Principals = servicePrincipal2,userPrincipal2
   ```

3. **Associate service principals with Agent Pools**

   A distributed platform can span multiple ports on multiple systems. Each agent pool defines where one part (JVM) or the platform will run. Each agent pool entry contains an alias, an IP Address, a port, and a service principal alias. The principal alias specifies what service principal this pool will be associated with. Add the service principal alias you defined above that you want to associate with your agent pool:

   ```
   #----------------------------------------------------------------------
   # Agent Pools (Java Virtual Machines)
   #----------------------------------------------------------------------
   AgentPool.1.Alias     = Pool1
   AgentPool.1.IpAddress = systemname.ibm.com
   AgentPool.1.Port      = 55551
   AgentPool.1.Principal = servicePrincipal1

   AgentPool.2.Alias     = Pool2
   AgentPool.2.IpAddress = systemname.ibm.com
   AgentPool.2.Port      = 55552
   ```

```
    AgentPool.2.Principal = servicePrincipal1

AgentPool.3.Alias     = Pool3
AgentPool.3.IpAddress = systemname.ibm.com
AgentPool.3.Port      = 55553
AgentPool.3.Principal = servicePrincipal2
```

4. **Define agent start-up authority**

   Define which users have the capability to start each of the agents defined on your secure platform.
   Add one or more user principal aliases to the EligiblePrincipal parameter:

```
#------------------------------------------------------------------------
# Permitted Agents
#------------------------------------------------------------------------
Agent.1.Alias=Agent1
Agent.1.AutonomyLevel=Medium
Agent.1.ClassName=com.ibm.able.platform.examples.EServerTemplateAgent
Agent.1.ConstructorArgs=String:AgentName1
Agent.1.EligiblePrincipals=userPrincipal1,userPrincipal2
Agent.1.EligibleAgentPools=Pool2,Pool3
Agent.1.InitArgs=
Agent.1.LastChangedDate=January 11, 2003 11:11am
Agent.1.Type=Tester1
Agent.1.Vendor=IBM1
Agent.1.Version=1.1
```

5. **Define the algorithm and provider**

   You need to define the algorithm and provider of the KeyPairs the platform will use. By default, the
   preferences file will contain the following setting:

```
#------------------------------------------------------------------------
# Cryptography parameters
#------------------------------------------------------------------------
CryptographyAlgorithm = DSA
CryptographyProvider  = IBMJCE
```

After you add the necessary security data to **ableplatform.preferences**, save your changes. Turning on
security for the platform once it is correctly configured is as simple as opening **able.preferences** that
defines your platform, and changing the Security property to **Security=on**. If you are running an
unsecured platform, you will need to end and restart the agent platform for security changes to take
effect.

   Related concepts

   "Start the agent platform"
   After you define the agent platform and optionally secure your platform, you will need to start all the
   Java Virtual Machines associated with your agent services using iSeries CL commands.

   Related tasks

   "Configure your platform to use Kerberos" on page 76
   The intelligent agent platform uses Kerberos principals to authenticate users and services throughout
   the agent platform. Kerberos protocol, developed by Massachusetts Institute of Technology, allows a
   principal (a user or service) to prove its identity to another service within an insecure network.

Start the agent platform:

After you define the agent platform and optionally secure your platform, you will need to start all the
Java Virtual Machines associated with your agent services using iSeries CL commands.

After you have set up the agent platform and optionally configured security, you need to start the agent
platform. Because the platform is made up of one or more Java Virtual Machines, to start the platform
you need to start all of the JVMs that make up the platform.

The following instructions provide information on how to start the agent platform on an iSeries server:

Start the agent platform on an iSeries server

The following commands handle the starting and stopping of an agent platform on an iSeries server: Start Agent Services (STRAGTSRV) and End Agent Services (ENDAGTSRV) commands.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

- Start Agent Services (STRAGTSRV) command

  When you run STRAGTSRV a separate JVM will be started for each pool on the system you are running the command from. This command starts a QAHASBMTER job that will find all of the JVMs (agent pools) that need to be started. When it discovers an agent pool, it will start a separate QAHASBMTEE job for each pool. Once the command has completed successfully, there should be a separate QAHAPLTFRM job in QSYSWRK with a status of SIGW for each pool. STRAGTSRV has the following keywords:

  PREFDIR
  > Sets the location of the following files: able.preferences and ableplatform.preferences. Leaving the PREFDIR parameter at the default value will start or end the platform defined by the ableplatform.preferences and able.preferences file in the **/QIBM/ProdData/OS400/able/** directory. Multiple platforms could be started on the same system by using the PREFDIR parameter to point to different directories. When doing this you must be careful that two platforms do not overlap at all by using the same ports on a system.

  HOMEDIR
  > Sets the location of the home directory.

  ClASSPATH
  > Allows you to add any additional classpath that each JVM should include. STRAGTSRV automatically sets the default classpath to:
  > ```
  > classpath=
  > /QIBM/ProdData/Java400/:/qibm/proddata/os400/able:
  > /qibm/proddata/os400/able/ableplatform.jar:
  > /qibm/proddata/os400/able/able.jar:
  > /qibm/proddata/os400/able/ablebeans.jar:
  > /qibm/proddata/os400/able/jas.jar:
  > /qibm/proddata/os400/able/Jlog.jar:
  > /qibm/proddata/os400/Java400/ext/ibmjgssiseriesprovider.jar:
  > /qibm/proddata/os400/jt400/lib/jt400Native.jar:
  > /qibm/proddata/os400/Java400/ext/db2_classes.jar:
  > /qibm/proddata/os400/able/auifw.jar:
  > ```

  SBMJOBUSER
  > Runs a JVM (agent pool) with a different profile than the current profile you are calling the command with:
  > > PoolIdentifier
  > > The pool that should run with a different profile.
  > > User Profile
  > > The profile you want to use to start the agent pool (PoolIdentifier).

- End Agent Services (ENDAGTSRV) command

  Ends all of the platform JVMs on this system that are specified as agent pools in the file **ableplatform.preferences**. This command starts a QAHAPLTEND job that will find and end all agent pools. ENDAGTSRV has the following keywords:

  PREFDIR
  > Sets the location of the following files: able.preferences and ableplatform.preferences. Leaving the PREFDIR parameter at the default value will start or end the platform defined by the ableplatform.preferences and able.preferences file in the **/QIBM/ProdData/OS400/able/** directory. Multiple platforms could be started on the same system by using the PREFDIR

parameter to point to different directories. When doing this you must be careful that two platforms do not overlap at all by using the same ports on a system.

Note: If you have trouble starting or ending the agent platform, you can turn on tracing for the startup programs by adding or setting the QAHA_TRACE system environment variable to '1'. This will create log files in QUSRSYS/QAAHALOG. Files named QSBR<job number>, QSBE<job number, and QEND<job number> will be created for each QAHASBMTER, QAHASBMTEE, and QAHAPLTEND job that has run.

Related tasks

"Configure your agent platform" on page 72
Provides a brief overview of the agent platform, and then provides detailed configuration steps for modifying the platform preferences file. Before you begin using the Intelligent Agents console in iSeries Navigator, you first need to configure the agent platform.

"Configure platform security" on page 78
Before you begin, ensure that you have configured your Kerberos key distribution center (KDC).

"Secure your agent environment" on page 75
It is strongly recommended that you use Kerberos user and service principals to authenticate users, agent pools, and agent services to one another on or across a secure platform or distributed platform.

Manage agents

Use the agent console to connect to your domain and begin managing your agents. Find out how to control the level of automation associated with your agents, and easily respond to requests and track agent history.

The Intelligent Agents console for iSeries is a powerful management tool that allows you to work with your agents, and ensure that they are behaving in a manner that meets your expectations. To display the Intelligent Agents node in iSeries Navigator, select **View** → **Intelligent Agents** from the main menu.
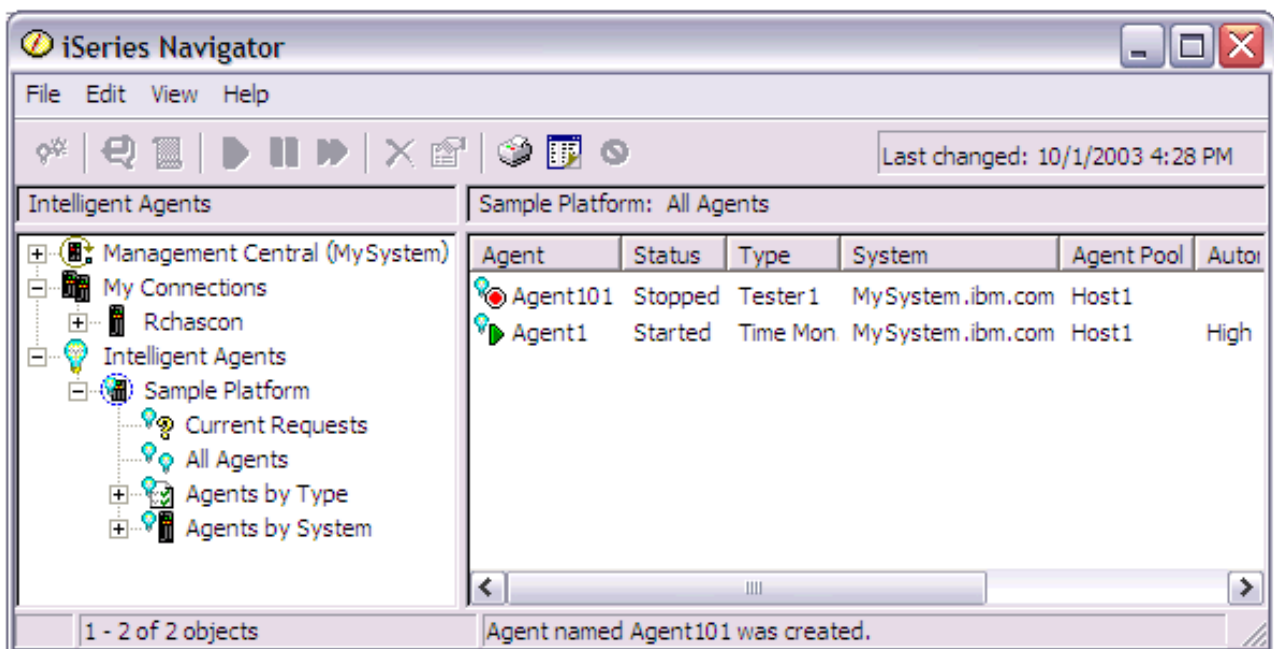

Figure 3. Working with agents in iSeries Navigator

After you set up your agent environment, you can begin working with the agent console by connecting to your host system (or systems) and creating an instance of an agent to run on that system. Use the

console to start, stop, suspend, delete, respond to, and view history of the agents running on your system or systems. You can also use the console to set up limitations on what actions an agent can perform automatically and what actions require permission.

Agent Automation:

The agent console gives you the capability to control and customize an agents behavior by associating a level of automation with that agent.

The Intelligent Agents console provides a way for you to control the automated actions an agent can take.

To view an agents capabilities, and change an agent's automation setting in iSeries Navigator, follow these steps:
1. Expand **Intelligent Agents**.
2. Expand your intelligent agents platform.
3. Select **All Agents**.
4. Right-click the agent you want to work with and select **Properties**
5. Select the **Automation** tab to display the agent's currently configured automation level.
6. Click **Capabilities** to display a list of the actions this agent can take, and the automation level associated with these capabilities.
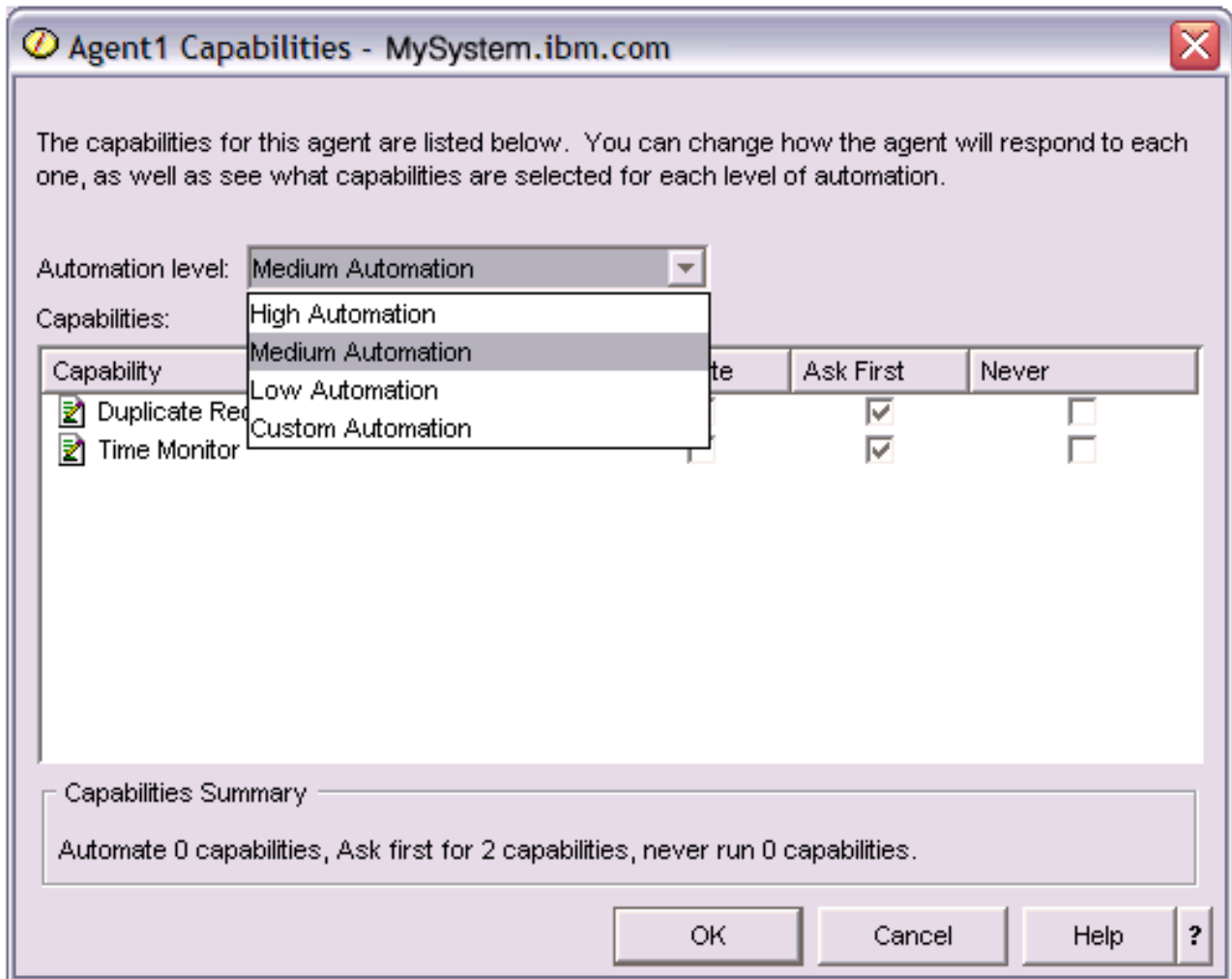
Figure 4. Viewing the automation level associated with the capabilities of a TimeMonitor agent

Every agent has a set of capabilities that define what kinds of actions that agent can perform. The agent console displays an agent's available capabilities associated with the agent's corresponding automation level. Each automation level setting (High automation, Medium automation, Low automation, and Custom automation) will change the states (Automate, Ask first, Never ask) of the available capabilities for the agent.

For example, If an agent has the capability to clear log files when full, when you change the level of automation from **High automation** to **Medium automation**, the agent's capability will change from a state of **Automate** to a state of **Ask first**. The agent will now request permission before it deletes a log file.

Specifying an agent's automation level will determine if an agent automatically performs an action, asks before performing an action, or never performs an action. The possible automation values are:

- **High automation**

  The agent will perform most actions automatically, but will ask before performing certain destructive actions. Depending on the agent, certain actions may require that the agent always request outside intervention before it performs that action, even when set to **High automation**.

- **Medium automation**

The agent will perform some actions automatically, and will ask before performing some actions. Depending on the agent, certain actions may require that the agent always request outside intervention before it performs that action, even when set to **Medium automation**.

- **Low automation**

  The agent will rarely perform any actions automatically. The agent will almost always request outside intervention before it performs any action.

- **Custom automation**

  The agent will automate, ask first, or never perform actions according to how the capabilities are manually configured.

Agent Communication:

Easily track and respond to agents that are requesting confirmation or permission to take action.

If the automation setting associated with an agent's capability is set to **Ask first**, before an agent performs an action, the agent will request a response from a user. Some agents will always request a response, regardless of their current automation setting. When an agent requests a response or is waiting to perform an action, the agent's Status field displays: **Needs response**.

To respond to an agent in iSeries Navigator:
1. Expand **Intelligent Agents**.
2. Expand your intelligent agents platform.
3. Select **All Agents**.
4. Right-click the agent and select **Respond...**.
5. Select the response you want to work with and click the **Respond** button.
6. The agent will display the problem it is currently seeking a response for. Select a response from the list of possible responses in the **Response** field, and click **OK**.
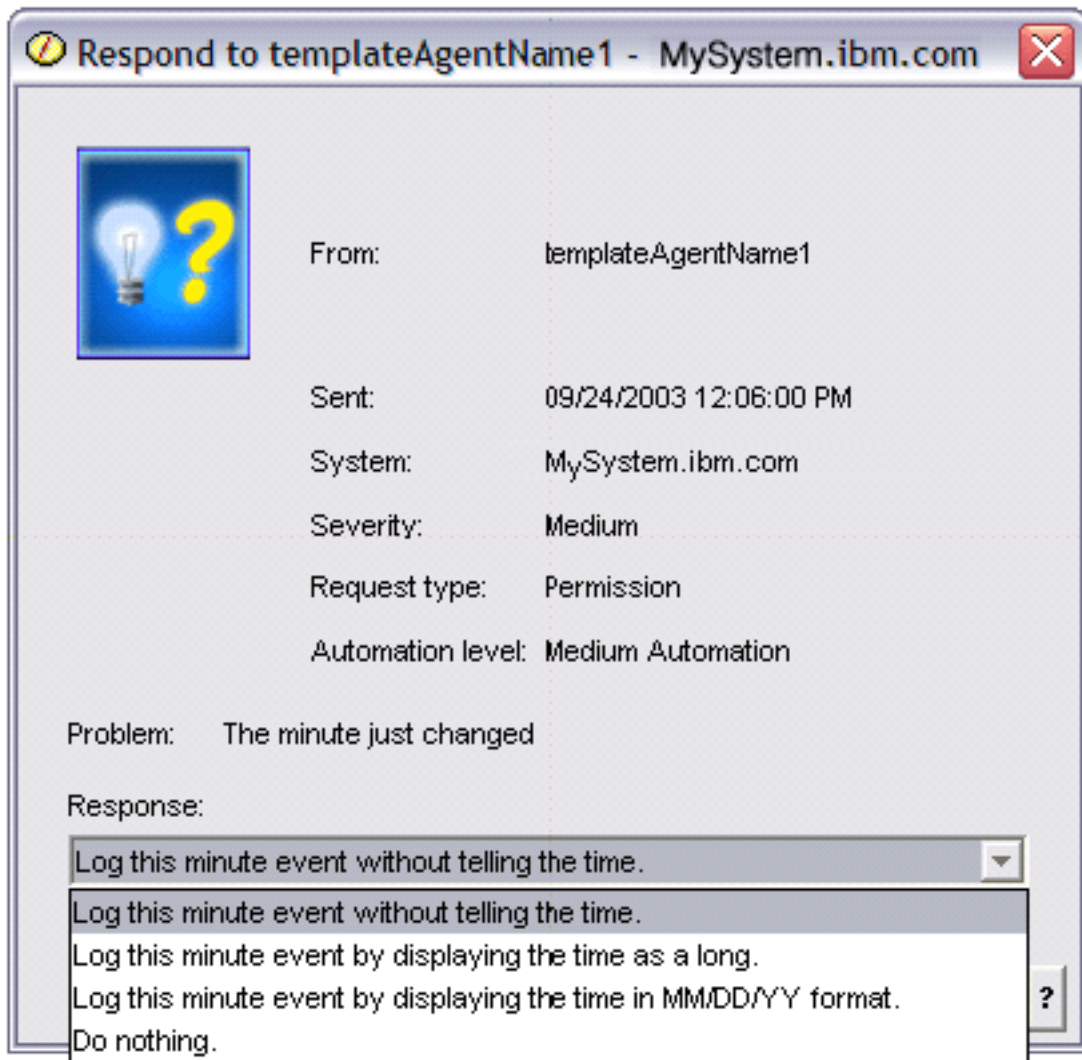
Figure 5. Responding to your agent's request

You can also view a list of all current requests by selecting **Current Requests** under the main **Intelligent Agents** menu.

Agent history:

The agent console logs a history of all your agents actions.

The agent console allows you to view the history of a agent's requests and actions. The history does not display current requests, only requests and actions that have been responded to. The history log is limited to 1000 entries, and will clear the oldest entry for each new entry that exceeds 1000.

To view an agent's history in iSeries Navigator, follow these steps:

1. Expand **Intelligent Agents**.
2. Expand your intelligent agents platform.
3. Select **All Agents**.
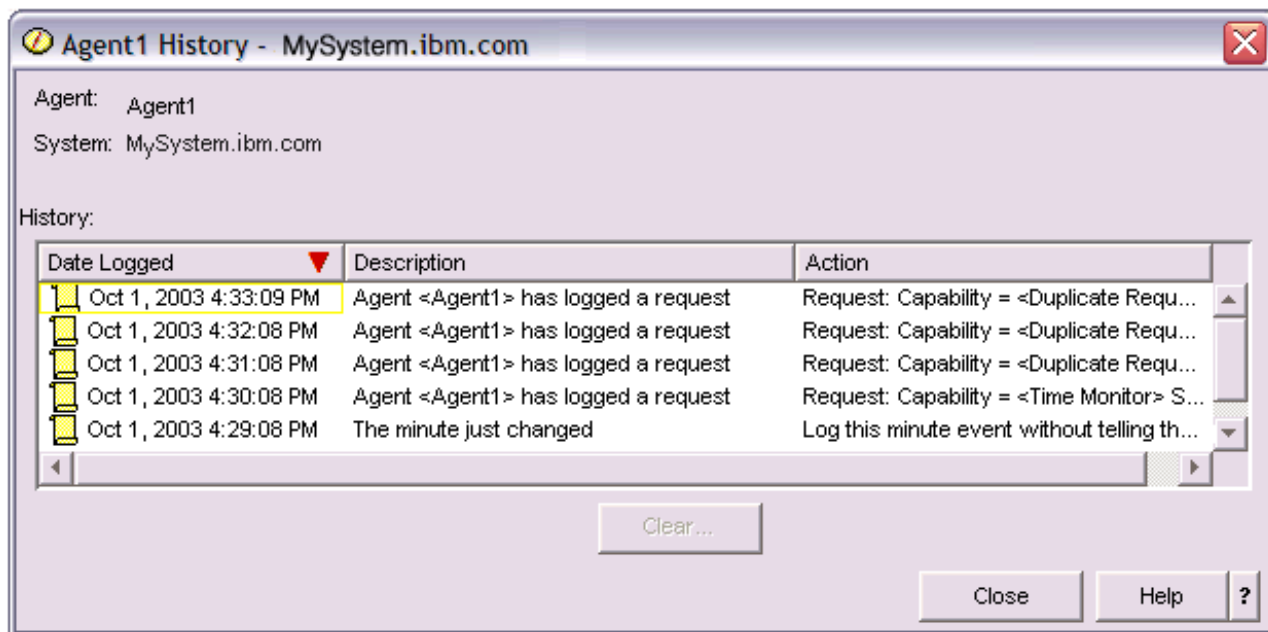4. Right-click the agent that you want to view history on, and select **History**.

Figure 6. Viewing the history of an agent's requests and actions

iSeries Navigator monitors

Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

The monitors included in iSeries Navigator use Collection Services data to track the elements of system performance of specific interest to you. Moreover, they can take specified actions when certain events, such as the percentage of CPU utilization or the status of a job, occur. You can use monitors to see and manage system performance as it happens across multiple systems and groups of systems.

With the monitors, you can start a monitor, and then turn to other tasks on your server, in iSeries Navigator, or on your PC. In fact, you could even turn your PC off. iSeries Navigator continues monitoring and performing any threshold commands or actions you specified. Your monitor runs until you stop it. You can also use monitors to manage performance remotely by accessing them with iSeries Navigator for Wireless.

iSeries Navigator provides the following types of monitors:

System monitor
> Collect and display performance data as it happens or up to 1 hour. Detailed graphs help you visualize what is going on with your servers as it happens. Choose from a variety of metrics (performance measurements) to pinpoint specific aspects of system performance. For example, if you are monitoring the average CPU utilization on your server, you can click any collection point on the graph to see a details chart that shows the 20 jobs with the highest CPU utilization. Then, you can right-click any of these jobs to work directly with the job.

Job monitor
> Monitor a job or a list of jobs based on job name, job user, job type, subsystem, or server type.

Choose from a variety of metrics to monitor the performance, status, or error messages for a job. To work directly with a job, just right-click the job from the list that is shown in the Job Monitor window.

Message monitor

Find out whether your application completes successfully or monitor for specific messages that are critical to your business needs. From the Message Monitor window, you can see the details of a message, reply to a message, send a message, and delete a message.

B2B activity monitor

If you have an application like Connect for iSeries configured, you can use a B2B activity monitor to monitor your B2B transactions. You can view a graph of active transactions over time, and you can run commands automatically when thresholds are triggered. You can search for and display a specific transaction as well as view a bar graph of the detailed steps of that specific transaction.

File monitor

Monitor one or more selected files for a specified text string, for a specified size, or for any modification to the file.

To find out more about monitors, see the following topics:

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"iSeries Navigator for Wireless" on page 143
iSeries Navigator for Wireless allows you to monitor performance data over a wireless connection, using a personal digital assistant (PDA), Internet-ready telephone, or traditional Web browser. The iSeries Navigator for Wireless uses the performance data collected by Collection Services.

Related reference

"Select a performance management strategy" on page 8
Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.

"Set up your environment to manage performance" on page 11
iSeries servers include powerful applications for managing system performance. However, they must be properly configured in order to meet the specific needs of your unique business environment. Learn how to configure applications to routinely collect, monitor, and analyze performance data.

"Collect information about system resource utilization" on page 16
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.

"Collect information about an application's performance" on page 17
An application might be performing slowly for a variety of reasons. You can use several of the tools included in i5/OS and other licensed programs to help you get more information.

"Display performance data" on page 20
After you have collected performance data, learn how to display the data using the most appropriate tool for your purposes.

"Network performance" on page 25
The network design, hardware resources and traffic pressure often have a significant effect on the performance of e-business applications. You can use this topic for information on how to optimize network performance, and tune server communication resources.

"Commands for i5/OS performance" on page 139
i5/OS includes several important functions to help you manage and maintain system performance.

Monitor concepts

Monitors can display real-time performance data. Additionally, they can continually monitor your system in order to run a selected command when a specified threshold is reached. Learn how monitors work, what they can monitor, and how they can respond to a given performance situation.

The system monitors display the data stored in the collection objects that are generated and maintained by Collection Services. The system monitors display data as it is collected, for up to one hour. To view longer periods of data, you should use Graph history. You can change the frequency of the data collection in the monitor properties, which overrides the settings in Collection Services.

You can use monitors to track and research many different elements of system performance and can have many different monitors running simultaneously. When used together, the monitors provide a sophisticated tool for observing and managing system performance. For example, when implementing a new interactive application, you might use a system monitor to prioritize a job's resource utilization, a job monitor to watch for and handle any problematic jobs, and a message monitor to alert you if a specified message occurs on any of your systems.

Setting thresholds and actions

When you create a new monitor, you can specify actions you want to occur when the system metric reaches a specified threshold level, or an event occurs. When threshold levels or events occur, you can choose to run an i5/OS command on the endpoint systems, such as sending a message or holding a job queue. Additionally, you may choose to have the monitor carry out several predefined actions such as updating the event log and alerting you by either sounding an alarm on your PC or launching the monitor. Finally, you can automatically reset the monitor by specifying a second threshold level which causes the monitor to resume normal activity when it is reached.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"Graph history" on page 96
Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

Configure a monitor

You can configure a monitor in iSeries Navigator. Use this topic to learn how to set up a monitor 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 real-time performance data from your endpoint systems. Creating a new monitor is a quick and easy process that begins at the **New Monitor** window:

1. In iSeries Navigator, expand Management Central, select **Monitors**, right-click **System**, and then select **New Monitor**.
2. Specify a monitor name. From the **New Monitor-General** page specify a name for your monitor. Provide a brief description so you can find the monitor in a list of monitors.
3. Select metrics. Use the **New Monitor-Metrics** page to select your metrics. You can monitor any number of metrics on any number of endpoint systems or system groups.
4. View and change your metric information. Use the **New Monitor-Metrics** page to edit the properties for each metric. You can edit the collection interval, maximum graphing value, and display time for each metric you select.
5. Set threshold commands. Use the **Thresholds** tab on the **Metrics** page to enable thresholds and specify commands to run on the endpoint system whenever thresholds are triggered or reset.

6. Set threshold actions. Use the **New Monitor-Actions** page to specify the actions you want to occur when a metric threshold is triggered or reset.

7. Select your systems and groups. Use the **New Monitor-Systems and Groups** page to select the endpoint systems or system groups where you want to start a monitor.

After you have created your monitor, right-click the monitor name and select **Start** to run the monitor and begin working with monitor graphs.

Monitor metrics:

To effectively monitor system performance, you must decide which aspects of system performance you want to monitor. Management Central offers a variety of performance measurements, known as **metrics**, to help you pinpoint different aspects of system performance.

The **Metrics** page in the **New Monitor** window allows you to view and change the metrics that you want to monitor. To access this page, select **Monitors**, right-click **System**, and then select **New Monitor**. Fill in the required fields, and then click the **Metrics** tab.

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 you can use in your monitor include the following:

Metric groups:	Metric description:
CPU Utilization	The percentage of available processing unit time consumed by jobs on your system. Choose from the following types of CPU Utilization metrics for use in your monitors:
	• CPU Utilization (Average)
	• CPU Utilization (Interactive Jobs)
	• CPU Utilization (Interactive Feature)
	• CPU Utilization (Database Capability)
	• CPU Utilization (Secondary Workloads)
	• CPU Utilization Basic (Average)
	To learn more about these metrics and how to use them, see the online help available on the **General** tab of the **New Monitor** window or the **Monitor Properties** window in iSeries Navigator.
Interactive Response Time (Average and Maximum)	The response time that interactive jobs experience on your system.
Transaction Rate (Average)	The number of transactions per second completed by all jobs on your system.
Transaction Rate (Interactive)	The number of transactions per second completed on your system by the following types of jobs:
	• Interactive
	• Multiple requester terminal (MRT)
	• System/36™ environment interactive
	• Pass-through
Batch Logical Database I/O	The average number of logical database input/output (I/O) operations currently performed by batch jobs on the system.

Metric groups:	Metric description:
Disk Arm Utilization (Average and Maximum)	The percentage of disk arm capacity currently used on your system during the time you collect the data.
Disk Storage (Average and Maximum)	The percentage of disk arm storage that is full on your system during the time you collect the data.
Disk IOP Utilization (Average and Maximum)	How busy the disk input/output processors (IOPs) are on your system during the time you collect the data.
Communications IOP Utilization (Maximum and Average)	How busy the communications input/output processors (IOPs) are on your system during the time you collect the data.
Communications Line Utilization (Average and Maximum)	The amount of data that was actually sent and received on all your system communication lines.
LAN Utilization (Maximum and Average)	The amount of data that was actually 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 occurring in all of the user pools on the system.

If you need more help, click the **Help** button on the **New Monitor-Metrics** window. Once you become familiar with the Management Central metrics, which metrics you select will depend on the information needs of your computing environment. After you have selected 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.

> **Related concepts**
>
> "Identify and resolve common performance problems" on page 14
> Many different performance problems often affect common areas of the iSeries server. Learn how to research and resolve problems in common areas, for example, backup and recovery.

Scenarios: iSeries Navigator monitors

Provides scenarios that show 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 iSeries Navigator provide a powerful set of tools for researching and managing system performance. For an overview of the types of monitors provided by iSeries Navigator, see iSeries Navigator monitors.

For detailed usage examples and sample configurations, see the following scenarios:

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 iSeries system has enough resources to meet the current demands of your users and business requirements. For your system, CPU utilization is a particularly 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 iSeries Navigator, expand **Management Central → Monitors**, right-click **System Monitor**, and select **New Monitor...**
2. On the **General** page, enter a name and description for this monitor.
3. Click the **Metrics** tab, and enter the following values:
    a. Select the **CPU Utilization Basic (Average)**, from the list of Available Metrics, and click **Add**. CPU Utilization Basic (Average) is now listed under Metrics to monitor, and the bottom portion of the window displays the settings for this metric.
    b. For **Collection interval**, specify how often you would like to collect this data. This will override the Collection Services setting. For this example, specify **30 seconds**.
    c. To change the scale for the vertical axis of the monitor's graph for this metric, change the **Maximum graphing value**. To change the scale for the horizontal axis of the graph for this metric, change the value for **Display time**.
    d. Click the **Threshold 1** tab for the metrics settings, and enter the following values to send an inquiry message if the CPU Utilization is greater than or equal to 80%:
        1) Select **Enable threshold.**
        2) For the threshold trigger value, specify **>= 80** (greater than or equal to 80 percent busy).
        3) For **Duration**, specify **1** interval.
        4) For the **i5/OS command**, specify the following:
            ```
            SNDMSG MSG('Warning,CPU...') TOUSR(*SYSOPR) MSGTYPE(*INQ)
            ```
        5) For the threshold reset value, specify **< 60** (less than 60 percent busy). This will reset the monitor when CPU utilization falls below 60%.
    e. Click the **Threshold 2** tab, and enter the following values to hold all the jobs in the QBATCH job queue when CPU utilization stays above 80% for five collection intervals:
        1) Select **Enable threshold**.
        2) For the threshold trigger value, specify **>= 80** (greater than or equal to 80 percent busy).
        3) For **Duration**, specify **5** intervals.
        4) For the **i5/OS command**, specify the following:
            ```
            HLDJOBQ JOBQ(QBATCH)
            ```
        5) For the threshold reset value, specify **< 60** (less than 60 percent busy). This will reset the monitor when CPU utilization falls below 60%.
        6) For **Duration**, specify **5** intervals.
        7) For the **i5/OS command**, specify the following:
            ```
            RLSJOBQ JOBQ(QBATCH)
            ```

This command releases the QBATCH job queue when CPU utilization stays below 60% for 5 collection intervals.

4. Click the **Actions** tab, and select **Log event** in both the **Trigger** and **Reset** columns. This action creates an entry in the event log when the thresholds are triggered and reset.

5. Click the **Systems and groups** tab to specify the systems and groups you want to monitor.

6. Click **OK** to save the monitor.

7. From the list of system monitors, right-click the new monitor and select **Start**.

Results

The new monitor displays the CPU utilization, with new data points being added every 30 seconds, according to the specified collection interval. The monitor automatically carries out the specified threshold actions, even if your PC is turned off, whenever CPU utilization reaches 80%.

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: Job monitor for CPU utilization:

See an example job monitor that tracks the CPU utilization of a specified job and alerts the job's owner if CPU utilization gets too high.

Situation

You are currently running a new application on your iSeries server, and you are concerned that some of the new interactive jobs are consuming an unacceptable amount of resources. You would like the owners of the offending jobs to be notified if their jobs ever consume too much of the CPU capacity.

You can set up a job monitor to watch for the jobs from the new application and send a message if a job consumes more than 30% of the CPU capacity.

Configuration example

To set up a job monitor, you need to define which jobs to watch for, what job attributes to watch for, and what the monitor should do when the specified job attributes are detected. To set up a job monitor that accomplishes this goal, complete the following steps:

1. In iSeries Navigator, expand **Management Central → Monitors**, right-click **Job monitor**, and select **New Monitor...**

2. On the **General** page, enter the following values:
   a. Specify a name and description for this monitor.
   b. On the **Jobs to monitor** tab, enter the following values:
      1) For the **Job name**, specify the name of the job you want to watch for (for example, MKWIDGET).
      2) Click **Add**.

3. Click the **Metrics** tab, and enter the following information:
   a. In the **Available metrics** list, expand **Summary Numeric Values**, select **CPU Percent Utilization**, and click **Add**.
   b. On the **Threshold 1** tab for the metrics settings, enter the following values:
      1) Select **Enable trigger**.
      2) For the threshold trigger value, specify **>= 30** (greater than or equal to 30 percent busy).
      3) For **Duration**, specify **1** interval.

4) For the **i5/OS trigger command**, specify the following:

```
SNDMSG MSG('Your job is exceeding 30% CPU capacity')
TOUSR(&OWNER)
```

5) Click **Enable reset**.

6) For the threshold reset value, specify **< 20** (less than 20 percent busy).

4. Click the **Collection Interval** tab, and select **15 seconds**. This will override the Collection Services setting.

5. Click the **Actions** tab, and select **Log event** in both the **Trigger** and **Reset** columns.

6. Click the **Servers and groups** tab, and select the servers and groups you want to monitor for this job.

7. Click **OK** to save the new monitor.

8. From the list of job monitors, right-click the new monitor and select **Start**.

Results

The new monitor checks the QINTER subsystem every 15 seconds, and if the job MKWIDGET is consuming more than 30 percent of the CPU, the monitor sends a message to the job's owner. The monitor resets when the job uses less than 20% CPU capacity.

Scenario: Job monitor with Advanced Job Scheduler notification:

See an example job monitor that sends an e-mail to an operator when the threshold limit of a job is exceeded.

Situation

You are currently running an application on your iSeries server, and you want to be notified if the CPU utilization reaches the specified threshold.

If the Advanced Job Scheduler is installed on the endpoint system, you can use the Send Distribution using JS (SNDDSTJS) command to notify someone by e-mail when the threshold is exceeded. For instance, you could specify that the notification escalate to the next person if the intended recipient does not respond by stopping the message. You could create on-call schedules and send the notification to only those people that are on-call. You can also send the notification to multiple e-mail addresses.

Job monitor configuration example

This example uses the SNDDSTJS command to send a message to a recipient named OPERATOR, which is a user-defined list of e-mail addresses. You can also specify an e-mail address instead of a recipient or both. To set up a job monitor that accomplishes this goal, complete the following steps:

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

1. In iSeries Navigator, expand **Management Central** → **Monitors**, right-click **Job monitor**, and select **New Monitor...**.

2. On the **General** page, enter the following values:

   a. Specify a name and description for this monitor.

   b. On the **Jobs to monitor** tab, enter the following values:

      1) For the **Job name**, specify the name of the job you want to watch for (for example, MKWIDGET).

      2) Click **Add**.

3. Click the **Metrics** tab, and enter the following information:

a. In the **Available metrics** list, expand **Summary Numeric Values**, select **CPU Percent Utilization**, and click **Add**.

b. On the **Threshold 1** tab for the metrics settings, enter the following values:

1) Select **Enable trigger**.

2) For the threshold trigger value, specify **>= 30** (greater than or equal to 30 percent busy).

3) For **Duration**, specify **1** interval.

4) For the **i5/OS trigger command**, specify the following:

```
SNDDSTJS RCP(OPERATOR) SUBJECT('Job monitor trigger') MSG('Job
&JOBNAME is still running!')
```

5) Click **Enable reset**.

6) For the threshold reset value, specify **< 20** (less than 20 percent busy).

4. Click the **Collection Interval** tab, and select **15 seconds**. This will override the Collection Services setting.

5. Click the **Actions** tab, and select **Log event** in both the **Trigger** and **Reset** columns.

6. Click the **Servers and groups** tab, and select the servers and groups you want to monitor for this job.

7. Click **OK** to save the new monitor.

8. From the list of job monitors, right-click the new monitor and select **Start**.

Message monitor configuration example

If you use a message monitor, you can send the message text to the recipient. Here is an example of a CL program that retrieves the message text and sends an e-mail to all on-call recipients with the SNDDSTJS command.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 146.

```
PGM PARM(&MSGKEY &TOMSGQ &TOLIB)

DCL &MSGKEY   *CHAR 4
DCL &TOMSGQ   *CHAR 10
DCL &TOLIB    *CHAR 10

DCL &MSGTXT   *CHAR 132

RCVMSG    MSGQ(&TOLIB/&TOMSGQ) MSGKEY(&MSGKEY)
          RMV(*NO)  MSG(&MSGTXT)
                 MONMSG CPF0000 EXEC(RETURN)

SNDDSTJS RCP(*ONCALL) SUBJECT('Message queue trigger')
MSG(&MSGTXT)
      MONMSG MSGID(CPF0000 IJS0000)

ENDPGM
```

This is the command that would call the CL program:

```
CALL SNDMAIL PARM('&MSGKEY' '&TOMSG' '&TOLIB')
```

Results

The monitor checks the QINTER subsystem every 15 seconds, and if the job MKWIDGET is consuming more than 30 percent of the CPU, the monitor sends an e-mail to the operator. The monitor resets when the job uses less than 20% CPU capacity.

See Work with notification for more information on the Advanced Job Scheduler notification function.

Related concepts

Work with notification

Scenario: Message monitor:

See an example message monitor that displays any inquiry messages for your message queue that occur on any of your iSeries servers. The monitor opens and displays the message as soon as it is detected.

Situation

You company has several iSeries servers running, and it is time-consuming to check your message queue for each system. 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 for your message queue that occur on any of your iSeries systems. The monitor opens and displays the message as soon as it is detected.

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 iSeries Navigator, expand **Management Central** → **Monitors**, right-click **Message monitor**, and select **New Monitor...**.
2. On the **General** page, enter a name and description for this monitor.
3. Click the **Messages** tab, and enter the following values:
   a. For **Message queue to monitor**, specify **QSYSOPR**.
   b. On the **Message set 1** tab, select **Inquiry** for **Type**, and click **Add**.
   c. Select **Trigger at the following message count**, and specify **1** message.
4. Click the **Collection Interval** tab, and select **15 seconds**.
5. Click the **Actions** tab, and select **Open monitor**.
6. Click the **Systems and groups** tab, and select the systems and groups you would like to monitor for inquiry messages.
7. Click **OK** to save the new monitor.
8. 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 on any of the iSeries servers that are monitored.

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 i5/OS commands when specified messages are received.

Graph history

Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

Graph history provides a graphical view of performance data collected over days, weeks, months, or years with Collection Services. You do not need to have a system monitor running to view performance data. As long as you use Collection Services to collect data, you can view the Graph History window.

Note: For more information about monitoring system performance, see the Track performance data topic.

Related concepts

"Track performance" on page 12
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.

"Monitor concepts" on page 89
Monitors can display real-time performance data. Additionally, they can continually monitor your
system in order to run a selected command when a specified threshold is reached. Learn how
monitors work, what they can monitor, and how they can respond to a given performance situation.

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for
iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the
graph history function. (If you prefer viewing real-time performance data, system monitors provide an
easy-to-use graphical interface for monitoring system performance.)

"View PM iSeries reports" on page 119
See examples of the PM iSeries reports and explanations of how to interpret those reports.

Related tasks

"Customize data collections" on page 39
When you use Collection Services to collect performance data, you control what data is collected and
how often it is collected.

"Manage collection objects" on page 51
When you use Collection Services to collect performance data, each collection is stored in a single
object.

"Activate PM iSeries" on page 102
PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

Related reference

"Select a performance management strategy" on page 8
Different business needs require different performance management strategies. Here are three basic
business models and their suggested performance management strategies.

"Display performance data" on page 20
After you have collected performance data, learn how to display the data using the most appropriate
tool for your purposes.

Graph history concepts

Contains a description of the available options for managing and displaying records of performance data.

Graph history displays data contained in the collection objects created by Collection Services. Therefore,
the type and amount of data available is dependent on your Collection Services configuration.

The amount of data that is available to be graphed is determined by the settings that you selected from
the Collection Services properties, specifically the collection retention period. Use iSeries Navigator to
activate PM iSeries over multiple systems. When you activate PM iSeries, you can use the graph history
function to see data that was collected days ago, weeks ago, or months ago. You go beyond the realtime
monitor capabilities, and have access to summary or detailed data. Without PM iSeries enabled, the graph
data field supports 1 to 7 days. With PM iSeries enabled, you define how long your management
collection objects remain on the system:

- **Detailed data** (attribute type *PFR in QMPGDATA.LIB or QPFRDATA.LIB)

  The length of time that management collection objects remain in the file system before they are deleted.
  You can select a specific time period in hours or days, or you can select **Permanent**. If you select
  Permanent, the management collection objects will not be automatically deleted.

- **Graph data** (attribute type *PFRDTL in QMGTC2.LIB)

The length of time that the details and properties data that is shown in the Graph History window remains in the system before it is deleted. If you do not start PM iSeries, you can specify one to seven days. If you do start PM iSeries, you can specify 1 to 30 days. The default is one hour.

- **Summary data** (attribute type *PFRHST in QMGTC2.LIB)

  The length of time that the data collection points of a graph can be displayed in the Graph History window or remain in the system before they are deleted. No details or properties data is available. You must start PM iSeries to enable the summary data fields. The default is one month. The summary data is summarized in one-hour intervals and does not support second- and third-level details.

- **Graph history status**

  The Graph History window now displays the graph history status. You also can re-create the graph history data if it is missing.

  Related concepts

  "Collection Services" on page 30
  Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

  Related tasks

  "Activate PM iSeries" on page 102
  PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

Use graph history

This topic contains step-by-step instructions to view graph history through iSeries Navigator.

Graph history is included in iSeries Navigator. To view the graph history of the data that you are monitoring with Collection Services, do these steps:

1. Follow the iSeries Navigator online help for starting Collection Services on either a single system or on a system group.
2. From the **Start Collection Services - General** page, select **Start IBM Performance Management for eServer iSeries** if needed.
3. Make changes to the other values for the collection retention period.
4. Click **OK**.
5. You can view the graph history by right-clicking either a system monitor or a Collection Services object and selecting **Graph History**.
6. Click **Refresh** to see the graphical view.

Tip: If the graph history data is missing, you can re-create it. To re-create the graph history data, right-click on the object in iSeries Navigator and choose **Create Graph History Data**.

Once you have launched a graph history, a window displays a series of graphed collection points. These collection points on the graph line are identified by three different graphics that correspond to the three levels of data that are available:

- A square collection point represents data that includes both the detailed information and properties information.
- A triangular collection point represents summarized data that contains detailed information.
- A circular collection point represents data that contains no detailed information or properties information.

The system adds data from the active collection object (*PFR attribute) to the *PFRDTL and *PFRHST collection objects when the following occurs:

- If the collection object properties is set to add graph data and summary data when cycled, the collection is cycled.

| • If the already cycled object is selected and the menu option to summarize the data is selected.
| • If a system monitor is running, then data is added to the *PFRDTL object only, as the system monitor
| is running.

> **Related concepts**
>
> "Collection Services" on page 30
> Use Collection Services to collect performance data for later analysis by the Performance Tools for
> iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the
> graph history function. (If you prefer viewing real-time performance data, system monitors provide an
> easy-to-use graphical interface for monitoring system performance.)

IBM Performance Management for eServer iSeries

PM iSeries automates the collection, archival, and analysis of system performance data and returns
reports to help you manage system resources and capacity. PM iSeries uses the performance data
collected by Collection Services.

IBM Performance Management for eServer iSeries (PM iSeries) is automated and self-managing, which
allows for easy use. PM iSeries automatically triggers Collection Services to gather the nonproprietary
performance and capacity data from your server and then sends the data to IBM. All collection sites are
network secure, and the time of the transfer is completely under your control. 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 PM reports provide the information that:
| • Allow you to plan for and manage system resources.
| • Help analyze key performance indicators.
| • Help in determining when peak workload volumes are occurring and provide you with the data
| needed to adjust the job scheduling accordingly.
| • Assist you with balancing the workload across logical partitions (LPARs).
| • Can be tailored to allow you to do problem analysis on an individual day.

The IBM Operational Support Services for PM iSeries offering includes a set of reports, graphs, and
profiles that help you maximize current application and hardware performance (by using performance
trend analysis).

| This offering, when used with Workload Estimator (WLE), 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. WLE can size a systems consolidation or evaluate the upgrading of an LPARed system, by
| having PM iSeries send the historical data for multiple systems or LPARs to WLE

PM iSeries uses less than 1 percent of your CPU). It uses approximately 58 MB of disk space, which
depends on your hardware model and the size of your collection intervals.

> **Related concepts**
>
> "Determine when and how to expand your system" on page 7
> 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.
>
> "Track performance" on page 12
> 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 tasks**

"Collecting performance data across partitions" on page 60
IBM Performance Management for @**server** iSeries (PM eServer iSeries) automatically triggers
Collection Services to gather nonproprietary performance and capacity data from your server and then
sends the data to IBM for analysis.

Related reference

"Select a performance management strategy" on page 8
Different business needs require different performance management strategies. Here are three basic
business models and their suggested performance management strategies.

"Set up your environment to manage performance" on page 11
iSeries servers include powerful applications for managing system performance. However, they must
be properly configured in order to meet the specific needs of your unique business environment.
Learn how to configure applications to routinely collect, monitor, and analyze performance data.

"Collect information about system resource utilization" on page 16
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.

"Display performance data" on page 20
After you have collected performance data, learn how to display the data using the most appropriate
tool for your purposes.

PM iSeries concepts

Learn about the functions and benefits PM iSeries can provide and about important implementation
considerations.

PM iSeries 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.

Benefits of PM iSeries:

PM iSeries can help make managing system resources and capacity planning significantly easier. Learn
more specific ways to utilize PM iSeries.

When you use the PM iSeries function, you receive these benefits:

*   **Helps you avoid unfortunate surprises.**

    You avoid disappointing surprises. You are in control over managing the growth and performance of
    your system, which means that you manage the system. Your system does not manage you.

*   **Saves you time.**

    You eliminate the labor intensive and expensive tasks of collecting and reporting performance data by
    doing it automatically. This benefit gives you the advantage of focusing your resources on managing
    your system and applications.

*   **Allows you to plan ahead for maximum efficiency.**

    You can proactively plan for the financial requirements to keep your system running at its peak
    efficiency.

*   **Provides easy to understand information.**

    You understand the information, which in turn makes it easy for you to present to senior management
    when asked the question, "Why do we need to upgrade?"

*   **Allows you to forecast the future.**

    You can make projections of your data processing growth based on factual trend information.

- **Allows you to identify system problems.**

  PM iSeries data enables you to identify performance bottlenecks.
- **Allows you to help estimate the size of your next upgrade.**

  You can upload PM iSeries data to the Workload Estimator for iSeries for sizing your next upgrade.

  Related information

  IBM eServer Workload Estimator

Operational Support Services for PM iSeries offering:

PM iSeries offers a wide range of options. Use this information to decide which combination of services best suits your needs.

You can receive your graphs and reports either electronically or in printed form. You can receive electronic graphs monthly. You receive printed graphs monthly or quarterly. The PM iSeries service fee varies depending on how often you choose to receive your performance information and your choice of format, electronically or in printed form. Some of these report options are free, and some are not. The marketing and services organizations in each country can give you details on the support that is available. Visit the PM eServer iSeries Web site for information on the free and fee options.

  Related concepts

  "View PM iSeries reports" on page 119

  See examples of the PM iSeries reports and explanations of how to interpret those reports.

  Related information

  PM eServer iSeries Web site

Data collection considerations for PM iSeries:

PM iSeries uses Collection Services to gather performance data. Learn how PM iSeries and Collection Services work together to provide the data you need.

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 iSeries and Collection Services, you need to be aware of the implications that can occur when you are using PM iSeries.

Here are some guidelines to help you define your collections when you are using PM iSeries:
- **Select the QMPGDATA library to store your data.**

  The **Location to store collections** field uses the default value /QSYS.LIB/QMPGDATA.LIB when PM iSeries is active. If you replace QMPGDATA with some other value, PM iSeries cycles the collection on the hour and changes it back to QMPGDATA. If you want to collect data into a different library, change where PM iSeries looks for data. Type **GO PM400** from the command line, select option 3 (Work with Customization), and change the library name.
- **Collect data continuously with Collection Services.**

  PM iSeries satisfies this requirement by collecting data 24 hours a day with Collection Services. PM iSeries collects performance data at 15-minute intervals. PM iSeries uses the 15-minute interval default, but does not change what the 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 data categories in the standard plus protocol profile correspond to the *ALL value for the DATA parameter on the Start Performance Monitor (STRPFRMON) command. If you change this to any other value, PM iSeries changes it back on the hour. This is true even if you select

Custom and include all categories. The change takes effect immediately. The collection does not cycle (unless required to do so for other reasons). This action is done to gather enough information for PM iSeries reports.

- **Avoid making interim changes to collection parameters when PM iSeries is active.**

  For example, when you activate PM iSeries, the **Create database files during collection** field is checked as the default value. If you change this, PM iSeries changes it back to the default value on the hour. The change takes effect immediately. The collection does not cycle (unless required to do so for other reasons).

- **Ending Collection Services.**

  You can end Collection Services at any time from iSeries Navigator. If you end Collection Services, the following considerations apply when PM iSeries is running:

  – The PM iSeries scheduler starts Collection Services at the beginning of the next hour.

  – Days with little or no data collected are not included in trend calculations. Therefore, you should not interrupt Collection Services often.

  Related concepts

  "Collection Services" on page 30
  Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

  Related tasks

  "Turn off PM iSeries momentarily" on page 118
  Learn how you can stop PM iSeries momentarily.

  "Customize PM iSeries" on page 112
  Now that you have set up your network, you may need to customize PM iSeries to fit your needs.

Configure PM iSeries

To start using PM iSeries, you need to activate it, set up a transmission method that allows you to send data and receive reports, and, finally, customize data collection and storage.

PM iSeries 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 iSeries collection process. PM iSeries creates the library during activation if the library does not already exist.

To begin using PM iSeries, you need to perform the following tasks:

Activate PM iSeries:

PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

You must start PM iSeries to take advantage of its data collecting capabilities. You can start PM iSeries by using any one of the following methods:

Use iSeries Navigator

Use iSeries Navigator to activate PM iSeries over multiple systems. When you activate PM iSeries, you can use the Graph History function to see data that was collected days ago, weeks ago, or months ago. You go beyond the real-time monitor capabilities. You have access to summary data or detailed data. Without PM iSeries enabled, the graph data field supports 1 to 7 days. With PM iSeries enabled, you choose the length of time to retain the data.

To start PM iSeries from iSeries Navigator, do the following steps:

1. In iSeries Navigator, expand the system where you want to start PM iSeries.
2. Expand **Configuration and Service**.
3. Right-click **Collection Services**.
4. Select **PM eServer iSeries**.
5. Select **Start**.
6. Select the systems on which you want to start PM iSeries.
7. Click **OK**.

Reply to message CPAB02A in the QSYSOPR message queue

When the QSYSWRK subsystem starts, this message asks if you want to activate PM iSeries.

1. From the character-based interface, reply with a G to the message in QSYSOPR "Do you want to activate PM eServer iSeries? (I G C)." QSYSOPR message queue receives the message that PM eServer iSeries is activated.
2. Update your contact information. Issue the **GO PM400** command and specify option 1.

Issue the Configure PM eServer iSeries (CFGPM400) command

From the character-based interface, you can issue the Configure PM eServer iSeries (CFGPM400) command.

You can proceed to the next step in the setup process, which is to determine which transmission method to use to send data to IBM.

For an overview of iSeries performance topics, see Performance.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"Graph history concepts" on page 97
Contains a description of the available options for managing and displaying records of performance data.

"Graph history" on page 96
Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

"Determine which PM iSeries transmission method to use" on page 104
Determine how you want to send data. You can either gather your data with the Management Central inventory function and send the data with Electronic Service Agent (Extreme Support) or you can have PM iSeries gather the data and send it over the SNA protocol.

Related tasks

"Collecting performance data across partitions" on page 60
IBM Performance Management for @**server** iSeries (PM eServer iSeries) automatically triggers Collection Services to gather nonproprietary performance and capacity data from your server and then sends the data to IBM for analysis.

"Send PM iSeries data with Service Agent over Extreme Support (Universal Connection)" on page 104
PM iSeries uses Collection Services to gather the nonproprietary performance and capacity data from your server. After you have collected this data, you can use Electronic Service Agent over Extreme Support to send the data to IBM.

"Gather PM iSeries performance data" on page 105
You can use Management Central to gather your PM iSeries performance data.

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.

Determine which PM iSeries transmission method to use:

Determine how you want to send data. You can either gather your data with the Management Central inventory function and send the data with Electronic Service Agent (Extreme Support) or you can have PM iSeries gather the data and send it over the SNA protocol.

Since V5R1, the PM iSeries transmission process has taken advantage of the network configuration that you perform with Management Central to set up a central system and endpoint systems. However, you can still use the character-based interface to configure PM iSeries. Choose which transmission method you want to use:

- Send data with the Electronic Service Agent over Extreme Support

  If you choose this transmission method, you need to configure PM iSeries to have data gathered by the Management Central inventory function. Perform this configuration for PM iSeries if your servers have V4R5 or later of the operating system installed (you must also have the Universal Connection fixes applied). You would choose this method if you use Extreme Support.

- Send data with SNA protocol

  If you choose this transmission method, you need to configure PM iSeries by using the character-based interface. PM iSeries gathers your data and transmits it by using SNA. Perform this configuration for PM iSeries if your servers have i5/OS V4R5 or earlier installed.

Once you have implemented which transmission method you want to use, you are ready to do the other tasks to Manage PM iSeries.

> **Related concepts**
>
> Management Central
>
> **Related tasks**
>
> "Activate PM iSeries" on page 102
> PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.
>
> **Related reference**
>
> "Manage PM iSeries" on page 115
> Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Send PM iSeries data with Service Agent over Extreme Support (Universal Connection):

PM iSeries uses Collection Services to gather the nonproprietary performance and capacity data from your server. After you have collected this data, you can use Electronic Service Agent over Extreme Support to send the data to IBM.

To take advantage of these capabilities, you must have V5R1 or V5R2 installed on your servers or V4R5 with the Universal Connection fixes applied. Here are the steps to follow to configure for PM iSeries:

1. Activate PM iSeries.

   You must start PM iSeries to take advantage of its data collecting capabilities.

2. Set up your Management Central network.

   Define which server is your central system and which servers are your endpoint systems. You can use this network hierarchy to send your data from your endpoint systems to a central location before sending the data to IBM.

3. Connect to IBM to transmit your data with the Universal Connection.

This is the connection that Management Central will use to transmit the PM iSeries data to IBM. In previous releases, you used the electronic customer support (ECS) connection that ran over SNA. When you use the Universal Connection, you can transmit your data over TCP/IP.

4. Gather PM iSeries performance data.

   Use the Management Central inventory function to gather your data.

5. Send your data to IBM.

   Use the Electronic Service Agent, which is available under Extreme Support in the Management Central hierarchy, to send your data to IBM. The Electronic Service Agent uses the Universal Connection.

You can also send data with the SNA protocol.

Once you have configured PM iSeries, you are ready to do the other tasks to manage PM iSeries.

Related concepts

Universal Connection

Related tasks

"Collecting performance data across partitions" on page 60
IBM Performance Management for @**server** iSeries (PM eServer iSeries) automatically triggers Collection Services to gather nonproprietary performance and capacity data from your server and then sends the data to IBM for analysis.

Electronic Service Agent over Extreme Support

"Activate PM iSeries" on page 102
PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

Set up your Management Central network

"Gather PM iSeries performance data"
You can use Management Central to gather your PM iSeries performance data.

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Gather PM iSeries performance data:

You can use Management Central to gather your PM iSeries performance data.

Perform the following tasks:
1. Activated PM iSeries
2. Configured the Universal Connection
3. Set up your Management Central network
4. Verified that the Electronic Service Agent is installed on your system or accessible from your system

To gather PM iSeries performance data on an endpoint system or system group, follow these steps:
1. In iSeries Navigator, expand **Management Central**.
2. Expand **Endpoint Systems** or **System Groups**.
3. Right-click an endpoint system or a system group, and select **Inventory**.
4. Select **Collect**.
5. Select one or more inventories to collect. In this case, you would select **PM iSeries performance data**.

6. If you want an action to run on the central system when collection completes, select the action from the list.
7. Click **OK** to start collecting the data immediately or click **Schedule** to specify when to collect the data.

Once you have configured your servers, you are ready to do the other tasks to manage PM iSeries.

Related tasks

"Send PM iSeries data with Service Agent over Extreme Support (Universal Connection)" on page 104
PM iSeries uses Collection Services to gather the nonproprietary performance and capacity data from your server. After you have collected this data, you can use Electronic Service Agent over Extreme Support to send the data to IBM.

"Activate PM iSeries" on page 102
PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

Configured the Universal Connection

Set up your Management Central network

Verified that the Electronic Service Agent is installed on your system or accessible from your system

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Sending data with SNA protocol:

If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.

PM iSeries asks you a series of questions about the configuration and use of your servers. The Configure PM eServer iSeries display asks you questions about how you want your servers to send and receive PM iSeries performance data. The first part of the process involves setting up your network. The second part asks you how you want to transmit your data. When you use the character-based interface, you can use a direct dial line to transmit data.

Follow these steps to send data with SNA:
1. Activate PM iSeries

   You must start PM iSeries to take advantage of its data collecting capabilities.
2. Select which network configuration you want to use.

   Determine which network configuration you will use to transmit data. Choose how you connect to IBM by using a direct dial line, an existing Internet Service Provider (ISP), or a virtual private network (VPN). If you want to use ISP or VPN, you must configure a Universal Connection.

   If you decide to use the direct dial line to report data to IBM, you have several choices as to how you configure your network. Select which configuration is appropriate for your network, and perform the steps outlined for that particular configuration from the Configure PM eServer iSeries display:

   - As a single server that sends its data directly to IBM.
   - As a host server, which means that you want your server to receive performance data from other servers (remote servers) and then forward the data to IBM. The host server cannot be at a release level that is earlier than other servers. In other words, the host server must be at the same release level or later than other servers.
   - As a remote server, which means that you can send performance data to a host server. You identify on the Configure PM eServer iSeries display that you need a remote server, and then use option 5 (Work with remote iSeries systems) from the PM eServer iSeries menu to define your remote servers.
3. Work with remote servers.

If you choose to set up your network for a host server, you need to identify those servers that will send their data to your host server. You can ignore this step if you are using a single server or a remote server.

4. Customize PM iSeries.

   After you have configured your network, you need to establish the global parameters for the operation of the PM iSeries software. You need to define the PM iSeries data telephone number if you would like to connect to IBM with a direct dial line.

Once you have configured your servers, you are ready to do the other tasks for managing PM iSeries.

Related concepts

Virtual private network (VPN)

Related tasks

"Send PM iSeries data with Service Agent over Extreme Support (Universal Connection)" on page 104
PM iSeries uses Collection Services to gather the nonproprietary performance and capacity data from your server. After you have collected this data, you can use Electronic Service Agent over Extreme Support to send the data to IBM.

"Activate PM iSeries" on page 102
PM iSeries ships with i5/OS, but you must activate it to use its collecting capabilities.

"Set a direct dial line for PM iSeries" on page 114
For most locations, PM iSeries tries to select the correct data telephone number for your location.

Configure a Universal Connection

"PM iSeries network for a single server"
A single server sends its data directly to IBM.

"PM iSeries network for a host server" on page 108
A host server receives performance data from other servers and then forwards the data to IBM .

"PM iSeries network for a remote server" on page 109
A remote server sends its performance data to a host server.

"Work with remote servers" on page 109
In some sites, a host server in a network sends the required performance data to IBM for processing.

"Customize PM iSeries" on page 112
Now that you have set up your network, you may need to customize PM iSeries to fit your needs.

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

PM iSeries 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 iSeries for a single server only if PM iSeries gathers data and transmits data over SNA. From the Configure PM eServer iSeries (CFGPM400) display on your server:

1. Type **CFGPM400** from the command line.
2. Specify *YES for the **Send performance data to IBM** field.
3. Specify *NO for the **Receive performance data** field.
4. Accept the default library of QMPGDATA.
5. If you specify *YES for Send performance data to IBM, you see additional information that indicates whether the appropriate communications objects exist. If the objects do not exist, PM iSeries creates the communications objects for you for transmission. Respond appropriately to the additional displays.
6. Type your company's contact information on the Work with Contact Information display.

If you decide that the single server setup is not what you want, you can choose another SNA configuration option.

Once you have configured your servers, you are ready to do the other tasks to manage PM iSeries.

Related tasks

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

PM iSeries 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 iSeries for a host server only if PM iSeries gathers data and transmits data over SNA:

1. From the Configure PM eServer iSeries display on your host server
   - Type **CFGPM400** from the command line.
   - Specify *YES for the **Send performance data to IBM** field.
   - Specify *YES for the **Receive performance data** field.
   - Accept the default library of QMPGDATA.
2. From the Work with Remote iSeries Systems display on your host server
   - Press F6 (Create) to identify which servers will send their data to your host server.
   - Complete the fields and press Enter.

Note: The following situation occurs only if PM iSeries gathers data and transmits data over SNA. If you have a network of systems, it is recommended that you use the Universal Connection and Management Central in iSeries Navigator to gather and transmit the data for those systems.

PM iSeries 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 primary server. This action minimizes the performance impact on the primary 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 primary server.

If you decide that the host server setup is not what you want, you can choose another SNA configuration option.

Once you have configured your servers, you are ready to do the other tasks to manage PM iSeries.

Related tasks

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

PM iSeries 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 iSeries for a remote server only if PM iSeries gathers data and transmits data over SNA. From the Configure PM eServer iSeries display (CFGPM400) on your remote server, do these steps:

1. Type **CFGPM400** from the command line.
2. Specify *NO for the **Send performance data to IBM** field.
3. Specify *NO for the **Receive performance data** field.
4. Accept the default library of QMPGDATA.

Note: If you have a network of systems, it is recommended that you use the inventory function of iSeries Navigator to gather your data and then transmit the data for those systems over the Universal Connection.

If you decide that the remote server setup is not what you want, you can choose another SNA configuration option.

Once you have configured your servers, you are ready to do the other tasks to manage PM iSeries.

> **Related tasks**
>
> "Sending data with SNA protocol" on page 106
> If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.
>
> **Related reference**
>
> "Manage PM iSeries" on page 115
> Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Work with remote servers:

In some sites, a host server in a network sends the required performance 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 iSeries Systems display enables you to define these other servers.

Notes:

1. You do not have to use this display if you are setting up your network as a remote server or as a single server. You perform this task only if PM iSeries gathers data and transmits data over SNA.
2. If you have a network of systems, it is recommended that you use the inventory function of iSeries Navigator to gather your data and then transmit the data for those systems over the Universal Connection.

Follow these steps to define your remote servers:

1. Type **GO PM400** from the command line.
2. Type a 5 (Work with remote iSeries systems) from the PM eServer iSeries 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 iSeries 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 eServer iSeries Remote Site Maintenance display or the Change Remote Site iSeries display. The remote location name must be unique between remote servers.

PM iSeries 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. To manually schedule the transmission of data, see the PM iSeries scheduler.

The PM iSeries 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 iSeries to use a specific line connection.

Related tasks

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.

"Schedule jobs with PM iSeries" on page 117
Learn how to schedule jobs with PM iSeries.

"Create a device description for PM iSeries" on page 111
You can create a device description for PM iSeries.

"Work with remote servers in a non-APPN network"
The primary server receives PM iSeries data from other servers and then sends the data to IBM. The remote server sends PM iSeries data to the primary server.

"Customize PM iSeries" on page 112
Now that you have set up your network, you may need to customize PM iSeries to fit your needs.

Work with remote servers in a non-APPN network:

The primary server receives PM iSeries data from other servers and then sends the data to IBM. The remote server sends PM iSeries data to the primary 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 iSeries gathers data and transmits data over SNA.

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 1. Remote system

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 primary server. At the command line, type CRTDEVAPPC. Press F4 to prompt for the parameters, and define the values with the following information:

Table 2. Primary server

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 3. Vary on remote system

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 eServer iSeries Menu to add Q1PRMxxx as a remote server. See Work with remote servers for instructions on how to add a remote server.

Now that you have finished configuring PM iSeries, see Manage PM iSeries for other tasks that you can perform with PM iSeries.

Related tasks

"Work with remote servers" on page 109
In some sites, a host server in a network sends the required performance data to IBM for processing.

"Create a device description for PM iSeries"
You can create a device description for PM iSeries.

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Create a device description for PM iSeries:

You can create a device description for PM iSeries.

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 iSeries gathers data and transmits data over SNA.

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

"Work with remote servers" on page 109

In some sites, a host server in a network sends the required performance data to IBM for processing.

"Work with remote servers in a non-APPN network" on page 110

The primary server receives PM iSeries data from other servers and then sends the data to IBM. The remote server sends PM iSeries data to the primary server.

Related reference

Create Controller Description (APPC) (CRTCTLAPPC) command

Change Controller Description (APPC) (CHGCTLAPPC) command

Display Controller Description (DSPCTLD) command

"Manage PM iSeries" on page 115

Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Customize PM iSeries:

Now that you have set up your network, you may need to customize PM iSeries to fit your needs.

The Work with PM eServer iSeries Customization display provides you with the ability to:

Establish global parameters for the operation of PM iSeries 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
- Performance data library
- Removal specifications

Define your PM iSeries data telephone number

Outside the United States and Canada, you must provide PM iSeries with the telephone number of the IBM location that will receive your data. For most locations, PM iSeries tries to select the correct telephone data number for your location when you initiate the configure PM iSeries process.

Vary a line off and on with PM iSeries

The PM eServer iSeries Line Control display allows PM iSeries to vary the line off, transmit the PM iSeries data, and then put the line back in the connect pending state.

To customize the global parameters, do the following steps:
1. Type **GO PM400** from the command line.
2. Type a 3 from the PM eServer iSeries menu to display the Work with PM eServer iSeries Customization display and press Enter.

If you are using Collection Services to gather your PM iSeries data, you should take into account some data collection considerations for PM iSeries.

See manage PM iSeries for other tasks that you can perform with PM iSeries.

> **Related concepts**
>
> "Data collection considerations for PM iSeries" on page 101
> PM iSeries uses Collection Services to gather performance data. Learn how PM iSeries and Collection Services work together to provide the data you need.
>
> **Related tasks**
>
> "Sending data with SNA protocol" on page 106
> If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme Support, you can still use the character-based interface to transmit data.
>
> "Work with remote servers" on page 109
> In some sites, a host server in a network sends the required performance data to IBM for processing.
>
> **Related reference**
>
> "Manage PM iSeries" on page 115
> Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Verify the PM iSeries data number:

If your server is using a direct dial connection to IBM , you must verify that the PM iSeries telephone number is correct. The telephone number must also contain the correct prefixes for your line.

Note: This is for SNA transmissions only.

To check the format of the telephone number of the electronic customer support line, do the following steps:
1. Type

```
DSPDTAARA DTAARA(QUSRSYS/QESTELE)
```

and press Enter.

2. Determine the connection number prefix found in offset 0. For example, if offset 0 is **'T9:1800xxxxxxx'** the prefix is **T9:**.

3. Type

```
DSPDTAARA DTAARA(QUSRSYS/Q1PGTELE)
```

and press Enter.

4. Offset 0 (zero) is the dialing string that will be used. (The other numbers will not be used.)

5. If you use your ECS line to order PTFs, you can compare the format in offset 0 (zero) to the format used for the ECS line, CALL QESPHONE, make a note of the string being used, and compare it to the value found in step 2.

   The telephone numbers will be different but the prefix should be the same (that is, SST9:1800..., SST:1800...and so forth).

If you need to change your telephone number, use the Change Data Area (CHGDTAARA) command:

Type **CHGDTAARA**, where DTAARA is Q1PGTELE, LIB is QUSRSYS, the substring starting position is *ALL, and the New value is 'SST:18005475497'

Note: The new value should be your dialing prefix, followed by 18005475497 for U.S.A and Canada.

Now that you have completed your PM iSeries configuration, see manage PM iSeries for the tasks that you can perform.

Related tasks

"Set a direct dial line for PM iSeries"
For most locations, PM iSeries tries to select the correct data telephone number for your location.

Related reference

"Manage PM iSeries" on page 115
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Set a direct dial line for PM iSeries:

For most locations, PM iSeries tries to select the correct data telephone number for your location.

You should always verify that the PM iSeries data telephone number is correct. If you do not have information that contains the PM iSeries data telephone number and the PM iSeries support number, contact your local IBM support personnel. They can provide you with the proper telephone numbers.

Note: This telephone number is not required if you are transmitting data through the Universal Connection. You need this number only if you are using the direct dial line.

To define the PM iSeries data telephone number or to change the number, do the following steps:

1. Type **GO PM400** from the command line.

2. Type a 3 from the PM eServer iSeries Menu to display the Work with PM eServer iSeries Customization display and press Enter.

3. On this display, scroll forward until you see the section of the display that shows you the telephone number fields.

4. Type the correct dialing sequence in the **IBM PM eServer iSeries phone number** field. For many IBM modems, you are required to use the colon (:) character for dial tone.

   Related tasks

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme
Support, you can still use the character-based interface to transmit data.

"Verify the PM iSeries data number" on page 113
If your server is using a direct dial connection to IBM , you must verify that the PM iSeries telephone
number is correct. The telephone number must also contain the correct prefixes for your line.

Vary a line off and on with PM iSeries:

Sometimes the line that PM iSeries uses is in the connect pending state. This state does not allow PM
iSeries to access the line to transmit data.

The PM eServer iSeries Line Control display allows PM iSeries to vary off the line, transmit the data, and
then put the line back in the connect pending state. When you use this display, you can change the PM
iSeries transmission task (Q1PCM1) to check for line status and vary off the appropriate line. Once the
transmission is complete, the same line is placed in a connect pending state.

Note: This task is necessary only if PM iSeries gathers data and transmits data over SNA.

To vary off and on a line, do the following steps:

1. Start the PM iSeries line monitoring function by typing **PMLINMON** from the command line. You
   should see the PM eServer iSeries Line Control display.
2. Read the warning that is shown on the first display and press Enter.
3. Define the line, controller, and device combinations that PM iSeries needs to vary off.
4. Use the prompt **Do you want PM eServer iSeries automatic line control active?** as a master control
   switch for the function. If you specify **YES**, the PM iSeries function is active. If you specify **NO**, the
   function is disabled.

   If you specify **NO**, you do not need to define the line control list again when you specify **YES**. You
   can vary off and on a line by specifying the line only. You can vary off and on a line, controller, and
   device by specifying all three descriptions.
5. Verify the line, controller, and device that you defined. Press Enter to see a summary of your choices.
6. Press Enter to confirm your choices or press F12 to return to the previous display to change your
   entries.

You can also set up PM iSeries line control by using the Configure PM eServer iSeries (CFGPM400)
command.

> **Related reference**
>
> "Manage PM iSeries"
> Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

Manage PM iSeries

Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

After you have set up your network to use PM iSeries, you can perform the following tasks:

> **Related concepts**
>
> "Determine which PM iSeries transmission method to use" on page 104
> Determine how you want to send data. You can either gather your data with the Management Central
> inventory function and send the data with Electronic Service Agent (Extreme Support) or you can
> have PM iSeries gather the data and send it over the SNA protocol.
>
> **Related tasks**

"Send PM iSeries data with Service Agent over Extreme Support (Universal Connection)" on page 104
PM iSeries uses Collection Services to gather the nonproprietary performance and capacity data from
your server. After you have collected this data, you can use Electronic Service Agent over Extreme
Support to send the data to IBM.

"Gather PM iSeries performance data" on page 105
You can use Management Central to gather your PM iSeries performance data.

"Sending data with SNA protocol" on page 106
If you choose not to take advantage of sending data with the Electronic Service Agent over Extreme
Support, you can still use the character-based interface to transmit data.

"PM iSeries network for a single server" on page 107
A single server sends its data directly to IBM.

"PM iSeries network for a host server" on page 108
A host server receives performance data from other servers and then forwards the data to IBM .

"PM iSeries network for a remote server" on page 109
A remote server sends its performance data to a host server.

"Work with remote servers in a non-APPN network" on page 110
The primary server receives PM iSeries data from other servers and then sends the data to IBM. The
remote server sends PM iSeries data to the primary server.

"Create a device description for PM iSeries" on page 111
You can create a device description for PM iSeries.

"Customize PM iSeries" on page 112
Now that you have set up your network, you may need to customize PM iSeries to fit your needs.

"Verify the PM iSeries data number" on page 113
If your server is using a direct dial connection to IBM , you must verify that the PM iSeries telephone
number is correct. The telephone number must also contain the correct prefixes for your line.

"Vary a line off and on with PM iSeries" on page 115
Sometimes the line that PM iSeries uses is in the connect pending state. This state does not allow PM
iSeries to access the line to transmit data.

Related reference
End PM eServer iSeries (Q1PENDPM) API

Deactivate PM iSeries:

Learn how you can stop PM iSeries.

To stop PM iSeries from running, you can use either of the following methods:

With iSeries Navigator

Perform the following steps:
1. In iSeries Navigator, expand the system where PM iSeries is running.
2. Expand **Configuration and Service**.
3. Right-click **Collection Services**.
4. Select **PM eServer iSeries**.
5. Select **Stop**.
6. Select the systems on which you want to stop PM iSeries.
7. Click **OK**.

With an API

Use the End PM eServer iSeries (Q1PENDPM) API to deactivate PM iSeries.

Change PM iSeries contact information:

Learn how to change your contact information from the original settings.

During the configuration of the PM iSeries 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 PM400** from the command line.
2. Type a 1 from the PM eServer iSeries Menu and press Enter. The Work with Contact Information display appears.
3. Change the contact information, as appropriate, and press Enter.

Schedule jobs with PM iSeries:

Learn how to schedule jobs with PM iSeries.

Integral to the PM iSeries software is a scheduler that automatically starts the jobs that are necessary to support the PM iSeries performance data collection and analysis.

Part of the PM iSeries 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 iSeries scheduled jobs, do the following:

1. Type **GO PM400** from the command line.
2. Type a 2 from the PM eServer iSeries 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 iSeries jobs.

PM iSeries scheduled jobs		
Job	Schedule	Function
Q1PTEST	At activation	Verifies that PM iSeries is activated and then goes to inactive status.
Q1PCM1	Weekly	Transmits the reduced performance data to IBM. This job is active only if you are using a direct dial line.
Q1PCM2	Daily	Varies communications offline.
Q1PPMSUB	Hourly	Ensures that Collection Services is collecting data.
Q1PDR	Daily	Performs data reduction and purges performance data.
Q1PPG	Monthly	Purges reduced performance data.
Q1PCM3	As needed	Varies communications offline after direct dial transmission fails to vary lines off.

PM iSeries scheduled jobs		
Job	Schedule	Function
Q1PCM4	As needed	Accesses the PM iSeries data from remote servers. This job is started only if you have added remote systems by using option 5 from the PM eServer iSeries Menu.
Q1PPMCHK	Every 4 hours	Verifies that data collection is active.
Q1PMONTH	Monthly	Allows for monthly transmission if you require an additional transmission during the month. The default value is set to inactive. This job is available only if you are using a direct dial line.

Related tasks

"Work with remote servers" on page 109
In some sites, a host server in a network sends the required performance data to IBM for processing.

"Turn off PM iSeries momentarily"
Learn how you can stop PM iSeries momentarily.

Omit items from PM iSeries analysis:

Learn how to omit jobs, users, and communications lines when performing an analysis with PM iSeries.

The PM iSeries software application summary includes an analysis of the top ten 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 top ten 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 PM400** from the command line.
2. Type a 4 from the PM eServer iSeries Menu and press Enter. The Work with Top Ten 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.

Turn off PM iSeries momentarily:

Learn how you can stop PM iSeries momentarily.

If you need to stop PM iSeries 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 PM400** 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 iSeries from verifying that Collection Services is collecting data. You must end what is currently being collected.

Note: PM iSeries will not start, cycle, or change Collection Services until the date and time to which you set the Q1PPMSUB job has been reached.

Related concepts

"Data collection considerations for PM iSeries" on page 101
PM iSeries uses Collection Services to gather performance data. Learn how PM iSeries and Collection Services work together to provide the data you need.

Related tasks

"Schedule jobs with PM iSeries" on page 117
Learn how to schedule jobs with PM iSeries.

Display PM iSeries status:

Learn how to use iSeries Navigator or the PM eServer iSeries menu to display PM iSeries status.

You can use either iSeries Navigator or the PM eServer iSeries Menu on your server to display the status of PM iSeries. Use the IBM Performance Management for eServer iSeries Status dialog to view the overall status of PM iSeries on one or more servers or groups. For example, you are shown details as to whether PM iSeries is active. Use the PM eServer iSeries Menu to view the Collection Services status, PM iSeries scheduler status, the performance data release, the last transmission attempt, performance data members, and the performance data size.

To view the overall status for PM iSeries from iSeries Navigator, do the following steps:
1. In iSeries Navigator, expand an endpoint system or a system group.
2. Expand **Configuration and Service**.
3. Right-click **Collection Services**.
4. Select **Performance Management eServer iSeries**.
5. Select **Status**.

To view the detailed status for PM iSeries from the PM eServer iSeries menu, do the following steps:
1. Type **GO PM400** from the command line.
2. Type a 6 from the command line and press Enter. See the online help for descriptions of each field.

View PM iSeries reports:

See examples of the PM iSeries reports and explanations of how to interpret those reports.

The output of the PM iSeries offering is a set of management reports and graphs on a monthly or quarterly basis. The PM iSeries offering has two options for the reports.

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.

Related concepts

"Operational Support Services for PM iSeries offering" on page 101
PM iSeries offers a wide range of options. Use this information to decide which combination of services best suits your needs.

"Graph history" on page 96
Graph history provides a graphical display of performance data collected by Collection Services over a specified period of time.

Related information

PM eServer iSeries Web site

PM iSeries reports

The iSeries server can be configured to send Collection Services data directly to IBM with PM iSeries. IBM then generates several reports that you can either view on the Web or have sent directly to you. Activating PM iSeries to automatically generate your reports not only saves you time and resources, but also allows you to plan ahead by forecasting your future growth needs.

The server automatically records various statistics about its operating environment during the course of normal operation. Collection Services has the capability to consolidate these statistics. PM iSeries can collect and transmit these statistics to IBM, which forms the basis for all PM iSeries reports that are produced. To produce these reports for viewing on the Web or for printing, PM iSeries must be activated and these statistics must be transmitted to IBM at least monthly or preferably more frequently.

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 each report and graph in detail and learn about some of their benefits and uses, visit the PM eServer iSeries Web site.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

Performance Tools

The Performance Tools licensed program includes many features to help you gather, analyze, and maintain system performance information. This includes assistance in managing performance over a distributed network, collecting and reporting on both summary and trace data, and capacity planning.

Performance Tools uses the performance data collected by Collection Services (sample data) and the trace data obtained from the Start Performance Trace (STRPFRTRC) command and the End Performance Trace (ENDPFRTRC) command.

The Performance Tools for iSeries licensed program allows you to analyze performance data in a variety of ways. Performance Tools is a collection of tools and commands for viewing, reporting, and graphing performance data. You can use Performance Tools for iSeries to view performance data collected with Collection Services or to view trace data collected with the Start Performance Trace (STRPFRTRC) command. You can then summarize the data into a report to research a performance problem on your system. You can also create graphs of the performance data to see resource utilization over time.

Performance Tools for iSeries contains a base product and two features (Manager and Agent). The base plus one of the features is required. For more information about the Manager and Agent features of Performance Tools, see the Manager and Agent feature comparison topic.

For more detailed information about how to use Performance Tools to collect data about the performance of a system, job, or program, look in the Performance Tools book. It also explains how to analyze and print data to help identify and correct any problems.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for

iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

"Manager and Agent feature comparison" on page 122
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 tasks

"Create database files from an existing collection object" on page 39
You can export performance data from an existing management collection object to database files.

Related reference

"Select a performance management strategy" on page 8
Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.

"Collect information about an application's performance" on page 17
An application might be performing slowly for a variety of reasons. You can use several of the tools included in i5/OS and other licensed programs to help you get more information.

Related information

Performance Tools PDF

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.

The Performance Tools for iSeries licensed program analyzes two distinct types of performance data: sample data and trace data. Collection Services collects sample data, which is summary data that is captured at regular time intervals. You collect sample data for trend analysis and performance analysis. The data relates to things such as storage pools and response times. However, Collection Services does not support the collection of trace data. Trace data is detailed data that you collect to gain additional information about specific jobs and transactions. To collect trace data, you can use either the Start Performance Trace (STRPFRTRC) command or performance explorer.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

Functions provided in Performance Tools:

Performance Tools 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.

Performance Tools includes reports, interactive commands, and other functions. For example, Performance Tools includes the following:

Tool	Description
Work with System Activity (WRKSYSACT) command	The Work with System Activity (WRKSYSACT) command allows you to work interactively with the jobs, threads and tasks currently running in the system. The WRKSYSACT command reports system resource utilization, including CPU utilization on a per-task basis for partitions that use a shared processing pool.

Tool	Description
Display Performance Data	The Display Performance Data graphical user interface allows you to view performance data, summarize the data into reports, display graphs to show trends, and analyze the details of your system performance all from within iSeries Navigator.
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 book.
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 book.
Performance explorer	Performance explorer is a data collection tool that helps you identify the causes of performance problems that cannot be identified by sample data that was collected by Collection Services or by doing general trend analysis. Use performance explorer for detailed application analysis at a program, procedure, module, or method level. For example, you can collect trace data on an individual program or procedure CPU and I/O statistics or individual object I/O characteristics. This tool is discussed in detail in the Performance Tools book.

Related concepts

"Performance explorer" on page 128
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 tasks

"Performance Tools plug-in" on page 123
You can view system resource utilization data in iSeries Navigator. You can view the data, graph it, and summarize it into reports. Find information about how to access this function here.

Related reference

Work with System Activity (WRKSYSACT) command

Related information

Performance Tools reports

Performance Tools PDF

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.

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.

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 Performance explorer reporting function and its associated commands are included in the base option in the Performance Tools for iSeries licensed program and, therefore, are available with either the Manager feature or the Agent feature. 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

"Performance explorer" on page 128
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 tasks

"Performance Tools plug-in"
You can view system resource utilization data in iSeries Navigator. You can view the data, graph it, and summarize it into reports. Find information about how to access this function here.

Related reference

"Performance Tools" on page 120
The Performance Tools licensed program includes many features to help you gather, analyze, and maintain system performance information. This includes assistance in managing performance over a distributed network, collecting and reporting on both summary and trace data, and capacity planning.

Performance Tools plug-in:

You can view system resource utilization data in iSeries Navigator. You can view the data, graph it, and summarize it into reports. Find information about how to access this function here.

Performance Tools can display performance data from the Display Performance Data graphical user interface (GUI), which is a plug-in for iSeries Navigator. From the GUI, you can view performance data, summarize the data into reports, display graphs to show trends, and analyze the details of your system performance.

Metrics

iSeries Navigator displays performance metrics over a selected interval of time. The performance metrics you can view in the Graphs pane of the Display Performance Data GUI include:
*   Transaction Count
*   Transaction Response Time
*   Total CPU Utilization
*   Interactive CPU Utilization
*   Batch CPU Utilization
*   Interactive Feature Utilization
*   High Disk Utilization
*   Machine Pool Page Faults/Second
*   User Pool Page Faults/Second
*   Exceptions

The Details pane allows you to view detailed performance data for the selected time interval in a variety of ways. To analyze your system performance, you can view job data, subsystem data, pool data, or disk unit data.

Reports

In addition to viewing graphs and detail data, you can also print reports from the Display Performance Data GUI. Performance reports allow you to research areas of the system that are causing performance problems. You can run different reports to see where your system resources are being used. Printing reports in Performance Tools is available only when option 1 (Manager feature) of Performance Tools for iSeries (5722-PT1) is installed on the central system.

The reports you can print from the Display Performance Data GUI include:
- System
- Component
- Job
- Pool
- Resource

Accessing through iSeries Navigator

The Display Performance Data GUI is a plug-in for iSeries Navigator. If you have already installed the plug-in, it can be accessed from iSeries Navigator by following these steps:

1. In iSeries Navigator, expand **My Connections** (or your active environment).
2. Expand the server that contains the performance data you want to view.
3. Expand **Configuration and Service**.
4. Right-click **Collection Services**, select **Performance Tools**, and then select **Performance Data**.
5. Select the performance data file that you want to display.
6. Click **Display**.

For more information on how to use the Display Performance Data GUI in iSeries Navigator, see the iSeries Navigator online help.

> **Related concepts**
>
> "Manager and Agent feature comparison" on page 122
> 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**
>
> "Display performance data" on page 20
> After you have collected performance data, learn how to display the data using the most appropriate tool for your purposes.
>
> "Functions provided in Performance Tools" on page 121
> Performance Tools 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.

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 and the capability to do dynamic configuration no longer worked with this methodology. 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.

In V5R3, Collection Services reports the total CPU that is consumed and the total CPU that is available to the partition within the interval. The concept of HVLPTASK and CPU scaling to whole virtual processors in shared processor environments does not exist. Collection Services no longer cycles the collection when the configuration changes.

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).

Release	Total CPU	Interactive	Batch	HVLPTASK
OS/400 V5R2 or earlier	50%	5%	10%	35%
OS/400® V5R3 or later	50%	17%	33%	N/A

Considerations

In V5R3, the Convert Performance Data (CVTPFRDTA) command performs normally. However, the data in the converted files is changed to be consistent with the unscaled system CPU data (QAPMSYSCPU database file). The results should be the same as if the data were collected on a V5R3 system, but the data is different than the values that existed in the files in a prior release.

The existing and unchanged tools that calculate CPU utilization do not show the correct results for shared processor partitions or partitions that have had configuration changes during data collection. This includes those tools that use the performance database as well as those that use the QPMLPFRD API.

You can copy a V5R3 management collection object (*MGTCOL) to a prior release and generate the database files. However, you should be aware of the following:
- The reported CPU data remains unscaled (shared processor environments). This means that the total system CPU that is reported by the tools using virtual processors (including Performance Tools) is not correct.
- A management collection object (*MGTCOL) that spans configuration changes will result in an inaccurate calculation of the percentage of CPU during those intervals after the change occurred.

   Related information

   Performance data files: QAPMSYSCPU

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 theQAPMSYSTEM 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 theQAPMSYSTEM 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 theQAPMSYSTEM 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 information

Performance data files: QAPMSYSTEM

Performance data files: QAPMSYSCPU

Performance Report columns

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 iSeries 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.

iSeries 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.

In V5R3, the Performance Tools licensed program was 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 iSeries 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.

Install and configure 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(5722PT1) 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(5722PT1) DEV(tape-device-name) OPTION(1)
  ```
- If you have purchased the Agent feature, use the following command:

  ```
  RSTLICPGM LICPGM(5722PT1) DEV(NAME) OPTION(2)
  ```

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(5722PT1) 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 information**
>
> Performance Tools PDF

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 collection functions and related commands of performance explorer are part of the i5/OS licensed program. The reporting function and its associated commands are part of the base option in the Performance Tools for iSeries licensed program and therefore, are available with either the Manager feature or the Agent feature. 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 concepts**
>
> "Set system benchmarks" on page 6
> Setting good system benchmarks will give you performance data for a properly tuned system. These performance benchmarks from both before and after system changes provide important information for both troubleshooting and planning.
>
> "User-defined transactions" on page 51
> Collection Services and performance explorer collect performance data that you define in your applications.
>
> "Manager and Agent feature comparison" on page 122
> 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**
>
> "Select a performance management strategy" on page 8
> Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.
>
> "Collect information about an application's performance" on page 17
> An application might be performing slowly for a variety of reasons. You can use several of the tools included in i5/OS and other licensed programs to help you get more information.
>
> "Functions provided in Performance Tools" on page 121
> Performance Tools 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.
>
> **Related information**
>
> AS/400 Performance Explorer Tips and Techniques book
>
> Performance Tools PDF

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 on an iSeries server. 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.
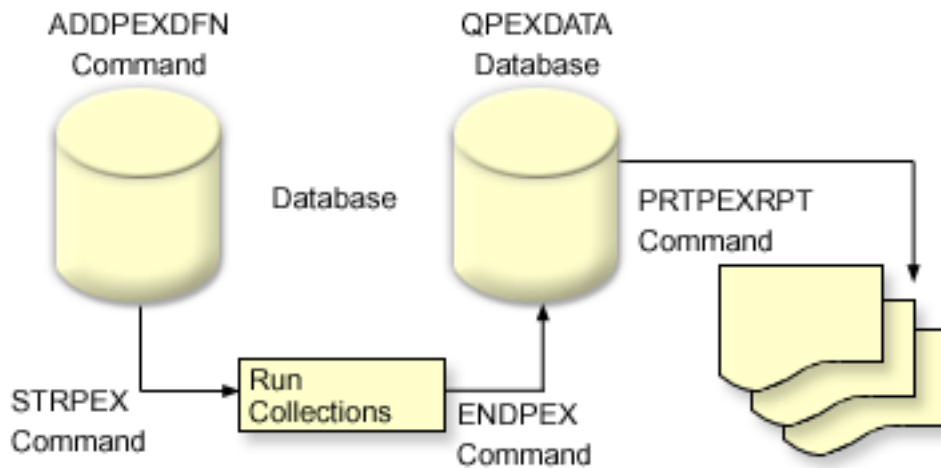
To learn more about performance explorer, refer to any of the following performance explorer topics.

Related concepts

"Collection Services" on page 30
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide an easy-to-use graphical interface for monitoring system performance.)

Related tasks

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 i5/OS 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, i5/OS 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" on page 135
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

"Configure performance explorer" on page 136
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) command

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(xxxxxxxxx)
```

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

Type of information contained in file	File name
Basic configuration information	QAYPECFGI
Communications event data	QAYPECMN
Disk event data	QAYPEDASD
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
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

Type of information contained in file	File name
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 i5/OS, 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" on page 131
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" on page 133
The data that performance explorer collects is stored in performance explorer database files.

Print Performance Explorer Report (PRTPEXRPT) command

Related information

Performance Tools PDF

Configure 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 iSeries server 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. 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, included in the Performance Tools licensed program, 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 iSeries. iDoctor 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.

   - Use the IBM Performance Trace Data Visualizer for iSeries (PTDV). PTDV is a Java application that you can use for performance analysis of applications. PTDV works with the Performance Explorer to allow you to view program flows and get details such as CPU time, wall clock time, number of cycles, and number of instructions, summarized by trace, job, thread, and procedures.

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 i5/OS operating system, except the Print Performance Explorer Report (PRTPEXRPT) command.

- The Performance Tools menu options.

    Related concepts

    "Performance explorer concepts" on page 129
    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 definitions" on page 131
    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

    Add PEX filter (ADDPEXFTR) command

    Start Performance Explorer (STRPEX) command

    Print Performance Explorer Report (PRTPEXRPT) command

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 erase data that was collected using this session name or RPLDTA(*YES) to add the
  collected data to the existing 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.

iDoctor for iSeries

The iDoctor for iSeries plugin consists of a variety of software tools for managing performance, for
example, PEX Analyzer for detailed trace data analysis and Job Watcher for trace-level information about
a job's behavior.

iDoctor for iSeries is a suite of tools consisting of these components: Consulting Services, Job Watcher,
Java Watcher, PEX Analyzer, and PTDV.

Consulting Services
    If you want expert consultants to analyze your system using one of the in-depth software tools
    from the iDoctor for iSeries Suite (PEX Analyzer or Job Watcher), select the Consulting Services
    component.

Job Watcher

Job Watcher displays real-time tables and graphical data that represent, in a very detailed way, what a job is doing and why it is not running. Job Watcher provides several different reports that provide detailed job statistics by interval. These statistics allow you to determine things like CPU utilization, DASD counters, waits, faults, call stack information, conflict information, and more.

Java Watcher

Java Watcher provides invaluable information to aid in debugging some of the most complex problems in the area of Java and WebSphere.

PEX Analyzer

PEX Analyzer evaluates the overall performance of your system and builds on what you have done with the Performance Tools licensed program. The Analyzer condenses volumes of trace data into reports that can be graphed or viewed to help isolate performance issues and reduce overall problem determination time. The Analyzer provides an easy-to-use graphical interface for analyzing CPU utilization, physical disk operations, logical disk input/output, data areas, and data queues. The Analyzer can also help you isolate the cause of application slowdowns.

PTDV The Performance Trace Data Visualizer for iSeries (PTDV) is a tool for processing, analyzing, and viewing Performance Explorer collection trace data residing in performance explorer database files. PTDV is a free component of iDoctor for iSeries.

Visit the iDoctor for iSeries Web site for more information.

Related concepts

"Find wait statistics for a job, task, or thread" on page 64
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).

Related reference

"Select a performance management strategy" on page 8
Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.

"Collect information about an application's performance" on page 17
An application might be performing slowly for a variety of reasons. You can use several of the tools included in i5/OS and other licensed programs to help you get more information.

Related information

"Scenario: Improve system performance after an upgrade or migration" on page 19
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.

Performance Trace Data Visualizer (PTDV)

Performance Trace Data Visualizer for iSeries (PTDV) is a Java application that can be used for performance analysis of applications running on iSeries.

PTDV works with performance explorer in the i5/OS base operating system to allow the analyst to view program flows and get details (such as CPU time, current system time, number of cycles, and number of instructions) summarized by trace, job, thread, and procedures. When visualizing Java application traces, additional details such as the number and type of objects created and information about Java locking behavior can be displayed. There is also support for performance explorer events generated by the WebSphere Application Server. PTDV allows sorting of columns, exporting of data, and many levels of data summarization.

For more information, go to the Performance Trace Data Visualizer Web site.

Related reference

Different business needs require different performance management strategies. Here are three basic
business models and their suggested performance management strategies.

An application might be performing slowly for a variety of reasons. You can use several of the tools
included in i5/OS and other licensed programs to help you get more information.

Performance Management APIs

Performance Management APIs provide services to manage collections. These APIs start, end, and cycle
collections, and they change and retrieve system parameters for the data collected. Many of the
Performance Management APIs use the performance data collected by Collection Services.

The performance management APIs allow you to collect and manage performance data using Collection
Services, performance collector, Performance Explorer, and PM iSeries.

The Performance Management APIs include:
- Collection Services APIs
- Performance Collector APIs
- Performance Explorer (PEX) APIs
- IBM Performance Management for eServer iSeries (PM iSeries) APIs

Commands for i5/OS performance

i5/OS includes several important functions to help you manage and maintain system performance.

These commands allow you to perform real-time monitoring of performance data from the
character-based interface. You can use these commands to answer specific questions about system
performance and to help you tune your system. For information about real-time monitoring from iSeries
Navigator, see iSeries Navigator monitors.

Command	Function
Work with Active Jobs (WRKACTJOB)	Allows you to review and change the attributes and resource utilization of the jobs running on your system.
Work with Disk Status (WRKDSKSTS)	Display the performance information and attributes for system disk units.
Work with System Status (WRKSYSSTS)	Provides an overview of current system activity. Specifically, it displays the number of jobs on the system and storage pool utilization information.
Work with System Activity (WRKSYSACT)	Work with jobs and tasks on your system. This command is part of the Performance Tools licensed program (PT1).
Work with Object Locks (WRKOBJLCK)	Work with and display locks on a specified object, including locks waiting to be applied.
Work with Shared Storage Pools (WRKSHRPOOL)	Display the utilization information and change attributes of shared storage pools, including machine and base pool.

Related reference

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.

"Display performance data" on page 20
After you have collected performance data, learn how to display the data using the most appropriate tool for your purposes.

"iSeries Navigator monitors" on page 87
Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

Extended Adaptive Cache

You can use Extended Adaptive Cache to improve system performance by collecting disk usage data, and then using those statistics to create a cache, effectively reducing the physical I/O requests for the disk.

Improve your iSeries system performance with Extended Adaptive Cache. Extended Adaptive Cache improves both the I/O subsystem and system response times by reducing the number of physical I/O requests that are read from disk. Extended Adaptive Cache generates statistical information for the data and then uses a mix of management strategies to determine which data to cache.

Note: In V5R4, Extended Adaptive Cache is supported on older iSeries systems. Extended Adaptive Cache uses the feature code, #4331 1.6 GB Read Cache.

Related concepts

"Tune performance" on page 21
When you have identified a performance problem, you will want to tune the system to fix it.

Extended Adaptive Cache Concepts

Explore Extended Adaptive Cache. Find information about planning, restrictions, and important considerations before you begin to use this tool.

Improve system performance with Extended Adaptive Cache, an advanced read cache technology that improves both the I/O subsystem and system response times by reducing the number of physical I/O requests that are read from disk. Extended Adaptive Cache not only improves the performance of database-read actions, but of all read actions. This includes read actions that are generated by other system components such as the Integrated xSeries® Server. It also works effectively in storage subsystems that have device parity protection or mirrored protection. Extended Adaptive Cache has proven to be highly effective on many types of workloads.

How the Extended Adaptive Cache works

Extended Adaptive Cache is integrated into the iSeries I/O subsystem. It operates at the disk subsystem controller level and does not affect the iSeries system processor. The storage I/O adapter manages the Extended Adaptive Cache by using a Read Cache Device (such as a solid state disk) to provide the cache memory.

Extended Adaptive Cache generates statistical information for the data, and then uses a mix of management strategies to determine which data to cache. The management of the cache is performed automatically within the I/O adapter and is designed to cache data by using a predictive algorithm. The algorithm considers how recently and how frequently the host has accessed a predetermined range of data.

The design of Extended Adaptive Cache was based on specific data management strategies of the iSeries server. Whether the disks are device parity protected, mirrored, or unprotected, the data stored on the disks has a tendency to occur in bands. This means that there are physically contiguous areas of disk storage where data is actively read, physically contiguous areas that are frequently written to, physically contiguous areas that are both actively read and written to, or physically contiguous areas of storage that are not frequently accessed.

This "banding" of data is accounted for in the Extended Adaptive Cache design. The goal is to cache bands characterized as read/write and read-only. A band that is characterized as write-only, while cached in the storage subsystem write cache, remains largely unaffected by Extended Adaptive Cache. Extended Adaptive Cache is also designed to not harm the performance of large blocks of data that are either sequentially written or sequentially read. In this instance, the pre-fetch capability of the disks, as well as other caches in the system, ensures a quick response time.

> **Related concepts**
>
> Device parity protection
>
> Mirrored protection

Restrictions and considerations for Extended Adaptive Cache:

See what components Extended Adaptive Cache requires and learn more about what to expect.

Before you begin using Extended Adaptive Cache, you should do some initial planning to take into account any restrictions or considerations that may pertain to your computing environment.

Restrictions

To use Extended Adaptive Cache, your system must have the following:
- One or more storage I/O adapters that support Extended Adaptive Cache (CCIN 2780 for systems running V5R2 or later).
- Performance Tools for iSeries licensed program for viewing the reported information.

Extended Adaptive Cache is automatically enabled on supported I/O adapters. There is no controlled on or off switch. Once the I/O adapter has been inserted into the subsystem, Extended Adaptive Cache is activated. It takes approximately an hour for Extended Adaptive Cache to monitor the data flow and populate the read cache memory. After an hour of running Extended Adaptive Cache, your system should show improved performance (depending on your current workload) and increased I/O throughput.

There are no restrictions for using Extended Adaptive Cache with regard to device parity protection and mirrored protection for other disks under the I/O adapter. Finally, Extended Adaptive Cache is designed specifically to complement iSeries Expert Cache, and may be used with or without it.

Considerations

Using the Extended Adaptive Cache allows you to attain a significant decrease in I/O response time and increase in system I/O throughput in most environments. As is the general case with caches, the system configuration and workload influence the effectiveness of Extended Adaptive Cache. Extended Adaptive Cache performs at the storage subsystem level. It caches data for the set of disks that are within that specific subsystem. Therefore, it is logical to add Extended Adaptive Cache to the most active and performance-critical storage subsystems within the system. Extended Adaptive Cache is not considered a pre-fetch type cache and therefore will not interfere with the read-ahead capabilities in the disk.

The larger the area of disk storage that is actively receiving I/O requests, the more selective Extended Adaptive Cache is about deciding when to bring new data into cache. This adaptive ability allows Extended Adaptive Cache to be effective on many workload types and sizes.

> **Related concepts**
>
> Device parity protection
>
> Mirrored protection
>
> To start Extended Adaptive Cache and increase your system's performance, purchase the Read Cache Device.

Start Extended Adaptive Cache:

To start Extended Adaptive Cache and increase your system's performance, purchase the Read Cache Device.

Once the Read Cache Device has been inserted into a disk slot on the subsystem, Extended Adaptive Cache will be activated. There is no user-controlled on or off switch. It takes approximately an hour for Extended Adaptive Cache to monitor the data flow and populate the Read Cache Device. After an hour of running Extended Adaptive Cache, your system should show improved performance (depending on your current workload) and increased I/O throughput.

> **Related concepts**
>
> "Restrictions and considerations for Extended Adaptive Cache" on page 141
> See what components Extended Adaptive Cache requires and learn more about what to expect.

Get Extended Adaptive Cache

After obtaining the performance data from Extended Adaptive Cache Simulator and deciding that you want Extended Adaptive Cache to improve your system's performance, you must purchase a Read Cache Device (RCD). Extended Adaptive Cache is automatically enabled through the RCD.

To begin using Extended Adaptive Cache, you must have:

- One or more storage I/O adapters that support Extended Adaptive Cache (CCIN 2748 for systems running V4R4 or later, or CCIN 2778 for systems running V5R1 or later, or CCIN 2757 for systems running the latest release of V5R2).
- A Read Cache Device for each storage I/O adapter that Extended Adaptive Cache is to be activated on (CCIN 6731 for systems running V4R4 or later).

Because Extended Adaptive Cache is automatically enabled through the RCD, there is no controlled on or off switch. The RCD may be added without system interruption through concurrent maintenance. The RCD resides in an internal disk slot and works with all other disk types and capacities. Be aware that all data in the Extended Adaptive Cache is also guaranteed to be on the disks. In the unlikely event of an RCD failure, there will be no data loss.

The Read Cache Device can be purchased wherever iSeries hardware is sold, or contact your local IBM representative.

> **Related concepts**
>
> Concurrent maintenance

Workload Estimator for iSeries

Helps you plan the size and timing requirements of your next upgrade. This tool is often used with PM iSeries to analyze trends in system performance and helps you efficiently manage the growth and expansion of your iSeries server.

The Workload Estimator helps you size system needs based on estimated workloads for specific workload types. PM iSeries is an integrated i5/OS function that users under processor warranty or on an IBM maintenance agreement can activate for no additional charge. In return, you receive capacity and performance analysis graphs useful in planning for and managing system growth and performance.

The Workload Estimator and PM iSeries have been enhanced to work with one another. Through a web-based application, you can size the upgrade to the required iSeries system that accommodates your existing system's utilization, performance, and growth as reported by PM iSeries. As an additional option, sizings can also include capacity for adding specific applications like Domino, Java, and WebSphere, or the consolidation of multiple AS/400 or iSeries traditional i5/OS workloads on one system. This capability allows you to plan for future system requirements based on existing utilization data coming from your own system.

"Determine when and how to expand your system" on page 7
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.

Related information

IBM eServer Workload Estimator

iSeries Navigator for Wireless

iSeries Navigator for Wireless allows you to monitor performance data over a wireless connection, using a personal digital assistant (PDA), Internet-ready telephone, or traditional Web browser. The iSeries Navigator for Wireless uses the performance data collected by Collection Services.

With your wireless device, you can:
* Run commands across multiple systems
* Start and view system, job, and message monitors
* Work with jobs and messages from the monitors (hold, release, end, reply, get details)
* Manage Integrated xSeries Server

For an overview of how iSeries Navigator for Wireless can help you get started with remote monitoring, see the topic iSeries Navigator for Wireless.

For complete and up-to-date information about remote monitoring, see the iSeries Navigator for Wireless home page.

Related concepts

iSeries Navigator for Wireless topic

Related reference

"iSeries Navigator monitors" on page 87
Monitors display current information about the performance of your systems. Additionally, you can use them to carry out predefined actions when a specific event occurs. You can use the system, message, job, file, and B2B transaction monitors to display and monitor information about your systems. The system and job monitors use the performance data collected by Collection Services.

PATROL for iSeries (AS/400) - Predict

PATROL for iSeries (AS/400) - Predict helps manage iSeries performance by automating many of the routine administration tasks required for high availability and optimal performance. Additionally, this product offers detailed capacity planning information to help you plan the growth of your iSeries environment.

The PATROL for iSeries (AS/400) - Predict product is a capacity planning tool that helps you estimate future iSeries requirements to accommodate transaction throughput and application workload increases. The estimation process is based on Collection Services data that provides resource utilization, performance, and 5250 online transaction processing (interactive) response time information that is measured on your iSeries server. The predictive analysis is performed through a graphical interface on a PC workstation.

For more information, refer to the BMC products Web site.

Related concepts

"Determine when and how to expand your system" on page 7
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.

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.

Use the following scenarios and configuration examples to learn more about managing performance.

Related information for iSeries Performance

Listed here are the product manuals and IBM Redbooks™ (in PDF format), Web sites, and information center topics that 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 as well as information about the Manager and Agent features.

- iSeries Performance Capabilities Reference

  This reference provides highly technical information about server performance useful for performance benchmarking, capacity planning, and planning for server performance.

IBM Redbooks

- 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 iSeries programmers, network and system management professionals, and other information technologists who are responsible for designing, developing, and deploying Web-based applications and information systems.

- AS/400 Performance Explorer Tips and Techniques

  This document provides descriptions and detailed examples of the performance explorer capabilities that were available for V3R6. Specific application examples and reports are provided.

- AS/400 Performance Management

  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.

- 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.

- IBM iDoctor iSeries Job Watcher: Advanced Performance Tool

    This document describes how to use Job Watcher functions, included with iDoctor for iSeries, to access detailed performance 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 with a specific focus on iSeries servers.

- 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.

- Managing OS/400 with Operations Navigator V5R1, Volume 1: Overview and More

    This volume presents an overview of Operations Navigator V5R1. It covers such things as managing jobs, subsystems, job queues, and memory pools; monitoring system performance metrics; jobs and messages; and Collection Services.

- Managing OS/400 with Operations Navigator V5R1, Volume 5: Performance Management

    This volume builds on the monitor, graph history, and Collection Services capabilities described in Volume 1. This book shows how to use these functions in an application environment.

Web sites

- iSeries Performance Capabilities Reference (www.ibm.com/servers/eserver/iseries/perfmgmt/resource.htm)

    This reference provides highly technical information about server performance useful for performance benchmarking, capacity planning, and planning for server performance.

- Three-In-One Benchmark (www.ibm.com/servers/eserver/iseries/hardware/threeinone)

    IBM developed a benchmark called the Three-in-One Benchmark to mirror the real-world demands facing IT companies. This report clearly demonstrates that the iSeries server is an excellent solution for today's small and medium businesses, which helps them run the applications they need without worrying about performance.

- Performance Management for IBM eServer iSeries (www-1.ibm.com/servers/eserver/iseries/perfmgmt/)

    Performance Management provides the capabilities for customers to understand and manage the performance of their computing environments. Read about the latest Performance Management functions and tools on this web site.

Saving PDF files

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

1. Right-click the PDF in your browser (right-click the link above).
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 installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

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.

Appendix. Notices

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

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

IBM Corporation

147

Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, 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 contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

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

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

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

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

Programming Interface Information

This Performance publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

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

| Advanced 36
| Advanced Function Printing
| Advanced Peer-to-Peer Networking
| AFP
| AIX
| AIX 5L
| AS/400
| DB2
| DB2 Universal Database
| Domino
| Electronic Service Agent
| Enterprise Storage Server
| eServer
| e(logo)server
| Hypervisor
| i5/OS
| IBM
| IBM (logo)
| iSeries
| Lotus
| NetServer
| OS/2
| OS/400
| POWER4
| POWER5
| PowerPC
| pSeries
| Redbooks
| System/36
| Virtualization Engine
| WebSphere
| xSeries

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 are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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.

IBM ®

Printed in USA