iSeries

Performance

Version 5 Release 3

iSeries

Performance

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 137.

Fifth Edition (August 2005)

This edition applies to version 5, release 3, modification 0 of IBM Operating System/400 (product number 5722-SS1) and IBM Performance Tools for iSeries (product number 5722-PT1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

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(TM) 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.

“What’s new in V5R3” on page 2
This topic describes what information is new or significantly changed in this release.

“Print this topic” on page 9
If you prefer a printed version of this information, go here to print the PDF.

“Plan for performance” on page 10
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.

“Set up your environment to manage performance” on page 14
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.

“Manage iSeries(TM) performance” on page 15
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.

“Applications for performance management” on page 29
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.

“Scenarios: Performance” on page 133
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. Find out about these examples.

“Related information” on page 133
IBM(R) related information contains technical, know-how, and “how to” information.

Note: This topic contains code examples. Read the "Code disclaimer information" on page 140 for important legal information.

What's new in V5R3

Here is what is new for this release:

- **"What's new: Monitors"**
  Find out about the new monitor support and the changes to the existing monitors.
- **"What's new: Collection Services" on page 3**
  Read about the changes to the performance database files and about the new Collection Services function.
- **"What's new: Performance Tools licensed program" on page 5**
  Read about the changes to the reports.
- **"What's new: PM iSeries" on page 8**
  Read about how PM eServer(TM) iSeries automatically collects data.
- **"What's new: Performance explorer" on page 9**
  Read about the changes to the performance explorer database files and about the new function.
- **Capacity on Demand**
  The Capacity on Demand topic has moved to Systems Management —> Capacity on Demand.
- **"Intelligent Agents" on page 65**
  iSeries(TM) 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.

What's new: Monitors

Here is what is new for this release:

You can use the new time zone system value (QTIMZON) to set your system time for a specific time zone. The Management Central monitors assume that the local system time (on the endpoint system) is configured correctly. If this does not occur, several metrics will never trigger. To ensure that your Management Central monitors process the correct local time, use the new time zone system value (QTIMZON) to ensure that the current time zone on your endpoint system matches your local time zone.

Now you can select to have thresholds for file monitors and message monitors automatically reset when your trigger command is run. When you define a threshold and specify a command to be run when the threshold is triggered, you just select **Automatically reset after trigger command has run**.

The CPU Utilization (Average) metric and CPU Utilization (Interactive Feature) metric changed their calculation methods to use the CPU utilization values instead of using the current calculation based on CPU count. In addition, the CPU Utilization (Average) metric can now go above 100% for partitions that support uncapped processors, so the associated range of the graph will also accommodate percentages greater than 100.

New fields were added to the system monitor metrics Disk Arm Utilization (Average), Disk Arm Utilization (Maximum), Disk Storage (Average), and Disk Storage (Maximum). These new fields provide information about a multipath disk unit, which is a unit that has multiple redundant paths from the system to the disk unit. Each path has a unique resource name.

- **Multipath unit** indicates that the resource represents a multipath disk unit.
- **Initial path of multipath unit** indicates that the resource represents the initial path of a multipath disk unit.
- **Production copy of a remotely mirrored independent ASP** indicates that the disk unit is in a production copy of a remotely mirrored independent ASP.

- **Mirror copy of a remotely mirrored independent ASP** indicates that the disk unit is in a mirror copy of a remotely mirrored independent ASP.

Job notification
You can use the notification function of the Advanced Job Scheduler to notify yourself, or others that you specify, if CPU utilization reaches a specified threshold. See, "Scenario: Job monitor with Advanced Job Scheduler notification" on page 92 for more about how this function works.

What's new: Collection Services

Here is what is new for this release:

CL commands

You now have the ability to start and end a performance collection with CL commands. You can also change the Collection Services properties with the CFGPFRCOL command. Use the CHKPFRCOL command to determine the current status of the Collection Services server job.

- Start Performance Collection (STRPFRCOL)
- End Performance Collection (ENDPFRCOL)
- Configure Performance Collection (CFGPFRCOL)
- Check Performance Collection (CHKPFRCOL)

New data and methodology for calculating CPU utilization

Read about the new method for "Reporting CPU utilization" on page 114 that is associated with partial processor partitions and dynamic configuration changes. This topic covers the following areas:

- The HVLPTASK is no longer reported by any system interface including the QAPMJOBx files.
- Collection Services no longer cycles when the configuration for the partition changes.
- The QAPMSYSCPU data is no longer scaled.
- What you see on prior releases if data is created from a V5R3 *MGTCOL management collection object.

Configured capacity

New support for logical partitions enables partitions to exceed their "Reporting configured capacity" on page 116 in situations where other partitions are not using all of their configured capacity.

Data port services

Collection Services now reports performance data obtained from data port services. This data can be helpful in understanding the performance of clients of data port services, such as remote independent ASP mirroring. When an independent ASP is mirrored remotely, data is written to the local independent ASP. The data is sent at the same time by using data port services to the corresponding independent ASP on the remote system. Performance on the local system could be affected negatively if the path to the remote independent ASP is slow. The data port services performance data provides information on the performance of this path. The data is recorded in the QAPMDPS file.

You can specify to collect data port services data from iSeries$^{(TM)}$ Navigator. A new category, DPS - Data port services, was added to the Available categories/Categories to collect field on the Collection Services dialog. The category is included in the *STANDARDP and *ENHCPCPLN profiles. You can also include the category in a custom profile.

Cross-partition performance data

For V5R3, Collection Services provides the framework to "Collecting performance data across partitions" on page 59 regardless of the operating system, for example, the partitions can be running OS/400(R), AIX(R) or Linux(TM). The IBM(R) Director Server component of IBM Director Multiplatform must be installed and running on the partition that is running Collection Services. IBM(R) Director Agent must be installed on the partitions from which you want to collect data. Collection Services collects the data from each partition, and PM iSeries summarizes the data. For information about the Linux operating systems that you can use for cross-partition performance data, see this Informational APAR: II13986.

You can collect logical partition data from iSeries Navigator. A new category, Logical partition, was added to the Available categories/Categories to collect field on the Collection Services dialog. The category is included in the *STANDARD, *STANDARDP, and *ENHCPCPLN profiles. You can also include the category in a custom profile.

Graph History

Prior to V5R3 you could obtain graph data in three ways. One, if graph data was available for a specific time range, it was returned. Two, if no graph data was available, but the raw data was available and the retention period was great enough for the graph data to be obtained from the raw data, then the raw data was dynamically converted to graph data and returned. Three, if no graph data was still available, the summarized data was returned. Beginning in V5R3, the dynamic conversion of raw data to graph data was eliminated. You can use the **Create graph data now** option to create your graph data.

Performance database files

The following table shows the new and changed database files.

Database file	Description
QAPMCONF	• New record keys that provide additional partition configuration information (SP). • New records added to provide the physical system view of the number of on demand processors, on demand processors available, on demand memory, and on demand memory available. • The interactive capacity granularity was increased from .1% to .01%. This change affects the records for GKEY IT and IL. • The record for GKEY PC now supports 255 partitions. Previously, it supported 99 partitions.
QAPMDISK	Collection Services now reports the following new disk performance data: indicates whether the disk unit is in a parity set (DSPS), indicates whether disk unit is in a high availability parity set (DSHAPS), indicates that the resource represents a multipath disk unit (DSMU), indicates that the resource represents the initial path of a multipath disk unit (DSIP), indicates that the disk unit is in a production copy of a remotely mirrored independent ASP (DSPC), and indicates that the disk unit is in a mirror copy of a remotely mirrored independent ASP (DSMC). In these cases, a multipath disk unit is a unit that has multiple redundant paths from the system to the disk unit. Each path has a unique resource name.
QAPMDPS	A new file that reports data port services performance data.

Database file	Description
QAPMIOPD	• Change to file description that states data is reported for Network Server (*IPCS category) and I/O adapters (*IOPBASE category). • New data type added. • New field added (XINWSD). • Change in description of INTSEC field. • Virtual I/O data was added to the file. For virtual I/O adapters on hosting partitions (partitions that provide the physical resources), data is provided about the I/O activity that occurs within this partition due to virtual device support that it provides on behalf of guest partitions.
QAPMJOBMI	New fields that support pages allocated (JBPGA), pages deallocated (JBPGD), and current user (JBCUSR).
QAPMJOBS	New fields that support pages allocated (JBPGA), pages deallocated (JBPGD), and current user (JBCUSR). New fields that support file system counters and journal counters. (These fields were added to QAPMJOBMI in the previous release.)
QAPMJSUM	A new job type, INF (Interactive feature) was added to the JSCBKT field.
QAPMLPAR	A new file that reports logical partition data if IBM Director Server licensed program is installed and running on the partition that is running Collection Services.
QAPMSYS	New fields added that report the data for a partition's availability and usage metrics.
QAPMSYSCPU	The individual CPU data that is reported in this file (SCPU01...32) changed for shared processor partitions. The data is no longer scaled for either capped or uncapped partitions. Previously, this data was scaled to match the configured whole virtual processors reported in field SCTNUM. A new field reports the current number of active processors (SCTACT).
QAPMSYSTEM	• The description for SMXDU was changed from maximum disk utilization to the largest utilization of all single path disk units and all paths of multipath disk units. • New fields added that report the data for a partition's availability and usage metrics. • New field, SYNUAL, added that reports the number of times that a noncached user authority lookup was performed. • The description for SYAUTH was changed from authority lookup count to object authority checks.

What's new: Performance Tools licensed program

This topic highlights changes to the Performance Tools licensed program for V5R3.

Batch and interactive CPU utilization

The process for "5250 online transaction processing (OLTP)" on page 117 is based upon a new bucket in QAPMJSUM that is provided by Collection Services. iSeries$^{(TM)}$ Access jobs can run either as batch or as interactive, but in previous releases all of the jobs were included in the CA4 bucket, which is charged to the Interactive CPU utilization column. A new category was added to the System Report, **iSeries Access - Batch** under the Non-interactive workload subsection. Additionally, the DDM server jobs were moved from the Interactive workload subsection to the Non-interactive workload subsection because they do not represent interactive workload.

New methodology for calculating CPU utilization
Collection Services uses new metrics that allow for better "Reporting CPU utilization" on page 114 by using shared processor pools. The new metrics are adjusted when configuration changes are applied to the partition. The new methodology eliminates the task of computing available CPU time. The concept of HVLPTASK and CPU scaling to whole virtual processors in shared processor environments does not exist anymore. The CPU values shown in Performance Tools represents the actual utilization in terms of processor units. Data that was collected in previous releases is converted by the Convert Performance Data (CVTPFRDTA) command to apply to the new methodology.

Component Report

- Virtual Processors (GKEY 13), Processor Units (GKEY PU), and Int Threshold (GKEY IT) metrics from the QAPMCONF file were added to the header section.
- A plus sign (+) was added next to the Unit column of the Disk Utilization section to identify multipath disk units. A letter H following the unit number indicates that the disk unit is in a high availability parity set. If no letter displays then the disk unit is a regular parity set.
- A column was added to the Component Interval Activity section to show the *amount of time exceeding the Interactive Threshold*, which is the time, in seconds, during which the interactive usage exceeds the configured Interactive Threshold. Amount of time exceeding the Interactive Threshold is a new field, **SYIFTE**, in the QAPMSYSTEM file. The *Interactive Threshold* is the percent of the total system CPU for interactive work. New this release, the interactive threshold value can change through the data collection period by reconfiguring the partition and is now in the **SYIFTA** field in the QAPMSYSTEM file.
- Dash signs (-) display under the High Disk and Unit columns when performance data is not available for the requested interval in the QAPMDISK file.
- A new column was added to the Component Interval Activity section to show the uncapped CPU time available for the system. This data is reported only for performance data that is collected on uncapped partitions.

System Report

- DDM server jobs were moved from the Interactive Workload subsection to the Non-Interactive Workload subsection.
- A new category for iSeries Access - Batch jobs was added to the Non-Interactive Workload section.
- Virtual Processors (GKEY 13), Processor Units (GKEY PU), and Int Threshold (GKEY IT) metrics from the QAPMCONF file were added to the header section.
- The DDM server job statistics are no longer shown in the first part of the Resource Utilization section. The first part of the Resource Utilization section continues to show statistics for Interactive jobs like Interactive, System/36$^{(TM)}$, MRT, iSeries Access, and Pass-through.
- The DDM server job statistics are no longer shown in the Interactive Resource Utilization Expansion subsection of the Resource Utilization Expansion section. The DDM server job statistics are now shown in the Non-Interactive Resource Utilization Expansion subsection.
- A plus sign (+) was added next to the Unit column of the Disk Utilization section to identify multipath disk units. A letter H following the unit number indicates that the disk unit is in a high availability parity set. If no letter displays then the disk unit is a regular parity set.
- Two rows were added to the Workload section. One to show information about the amount of time exceeding the Interactive Threshold and the other to show the shared processors pool utilization in terms of a percentage. The shared processor pool utilization row displays only for performance data that is collected on partitions using a shared processor pool.

Miscellaneous reports

- A plus sign (+) was added next to the Unit column of the Disk Utilization section to identify multipath disk units. A letter H following the unit number indicates that the disk unit is in a high availability parity set. If no letter displays then the disk unit is a regular parity set.
- A column was added to the Interactive Job Detail and Non-Interactive Job Detail sections of the Job Interval Report to show the current user of the job collected by Collection Services in the QAPMJOBMI file.

Performance advisor

Enhancements include changes to existing metrics to give more accurate recommendations and to add new metrics to analyze and give recommendations related to newer functions.

- Provide recommendations for Interactive Feature Utilization, Point-to-Point Protocol (PPP) activity, and TCP/IP activity.
- Update guidelines for CPU utilization, system metrics, and disk service time.

Display Performance Data display

- A new column was added to the Select Time Intervals to Display display to show the Interactive Feature Utilization (Int Feat Util). Dash signs (-) display under the High Dsk and Unit columns when performance data is not available for the requested interval in the QAPMDISK file.
- A field was added to the Display Performance Data display to show the percent of the processing capacity that is assigned for interactive work (Int Threshold). This information is taken from the IT field of the QAPMCONF file.
- Two fields were added to the Display Performance Data display to show the Virtual Processors (GKEY 13) and Processor Units (GKEY PU) metrics from the QAPMCONF file.
- A row was added to the Display Performance Data display to show the percentage of interactive capacity used by the system (Interactive Feature Utilization).
- A row was added to the Display Performance Data display to show the interactive CPU time in seconds over the threshold (Time exceeding Int CPU Threshold (in seconds)).
- Option 6 (Wait detail) was added to the Display Jobs display, which shows the wait time statistics, in seconds, for the requested job or task.

Analyze Performance Data display

- A new column was added to the Select Time Intervals to Analyze display to show the Interactive Feature Utilization (Int Feat Util). Dash signs are displayed under High Dsk and Unit columns when performance data is not available for the requested interval in the QAPMDISK file.
- A field was added to the Display Recommendations display to show the percent of the processing capacity that is assigned for interactive work (Int Threshold). This information is taken from the IT field of the QAPMCONF file.
- Two fields were added to the Display Recommendations display to show the Virtual Processors (GKEY 13) and Processor Units (GKEY PU) metrics from the QAPMCONF file.

Perform menu

- The Collection Services options from the Perform menu, Option 2 (Collect performance data), now take advantage of the new performance collection CL commands. Previously, these options used the collector APIs. These options now use the following CL commands:
  - Option 1 (Start Performance Collection): Start Performance Collection (STRPFRCOL)
  - Option 2 (Configure Performance Collection): Configure Performance Collection (CFGPFRCOL)
  - Option 3 (End Performance Collection): End Performance Collection (ENDPFRCOL)
- A new column was added to the Select Time Intervals display to show the Interactive Feature Utilization (Int Feat Util). Dash signs (-) are displayed under the High Disk and Unit columns when performance data is not available for the requested interval in the QAPMDISK file.

Work with System Activity (WRKSYSACT) command
This command was changed to calculate and show values consistent with the other performance tools. The CPU utilization in WRKSYSACT does not scale to whole virtual processors anymore. The HVLPTASK job is excluded from the list of tasks that consume CPU. In addition, WRKSYSACT was enhanced to show CPU values higher than 100% for uncapped processors, instead of capping the CPU utilization to 99.9%

Performance Tools plug-in for iSeries Navigator

- High Disk Utilization graph now excludes intervals with no disk information.
- The User Pool Faults/Second graph was enhanced to graph more than one pool, with one line per pool. There are two options for the User Pool Faults/second graph, which you can access by selecting **Preferences** from the Graphs menu. The options are:
  - Graph the top 10 pools with the highest fault rate, depending on the time period you select.
  - Graph specific pools that you select.

  This enhancement helps performance analysts view the impact that changes in the size of pools has on fault rates and also gives them a better understanding of pool activity in general.
- A new graph was added to show the Interactive Feature Utilization metric when it is provided by Collection Services. Interactive Feature Utilization, along with Time Exceeding Interactive CPU Threshold metrics will display in the Summary window and the Interactive Threshold to Data Properties page if the information is available in the collection.
- Library and member name added to Display Performance Data window title to accommodate the situation where more than one window is opened.
- A new metric, Uncapped CPU% Available, was added to the Total CPU Utilization graph.

What's new: PM iSeries

Here is what is new for this release:

The Performance Management/400 name was changed to IBM(R) Performance Management for eserver iSeries(TM) (PM eserver iSeries or PM iSeries). PM iSeries was chosen to better reflect the functions in the reports and tools that have changed over recent years to support the new functions in the iSeries.

The Universal Connection replaces IBM Global Network(R) (IGN) support for transmitting data.

Prior to V5R3, you could choose to omit the HvLp* job from the Omit Jobs from Top Ten display. In V5R3 the HvLp* job is no longer included when you do an upgrade. This change does not affect the data when you perform a migration.

Cross-partition performance data

For V5R3, Collection Services provides the framework to collect data from a partition regardless of the operating system, for example, the partitions can be running AIX(R) or Linux(TM). The IBM Director Server licensed program must be installed and running on the partition that is running Collection Services. Direction agents must be installed on the partitions. Collection Services collects the data from each partition, and PM iSeries summarizes the data.

You can specify to collect logical partition data from iSeries Navigator. A new category, Logical partition, was added to the Available categories/Categories to collect field on the Collection Services dialog. The category is included in the *STANDARD, *STANDARDP, and *ENHCPCPLN profiles. You can also include the category in a custom profile.

For the most current information about the PM iSeries reports, go to the PM eServer iSeries Web site .

What’s new: Performance explorer

Here is what is new for this release:

CL commands

Add PEX Definition (ADDPEXDFN)

- Add threads/tasks option (ADDTHDOPT)
  Specifies what types of threads and tasks should be included in the Performance Explorer session based on the creation time of the threads and tasks relative to the start time of the Performance Explorer session.
- Added Randomize element to INTERVAL parameter. You can specify *FIXED or *VARY values.
- Added Event format element to Base events (BASEVT) parameter and Communications events (CMNEVT) parameter
  The event format describes what data is collected for this event. *FORMAT1 provides the data used for most data analysis. The other formats allow for collection of other data related to these events. Values other than *FORMAT1 are valid for only the *PMCO and *SWOQ events. For all other events, *FORMAT1 will be used regardless of what format is specified.
- Added Save/Restore events (SAVRSTEVT) parameter.
  Specifies which save/restore events are included in the definition.

Add PEX Filter (ADDPEXFTR)

- Added Java(TM) trigger parameter (JVATRG)
  If a Java method entry event (*JVAENTRY) occurs that matches this trigger specification, then performance explorer collects all events specified in the performance explorer definition used for the active performance explorer session. The events are collected only for the thread where the trigger occurs.
- Added Java class filter parameter (JVACLSFTR)
  Specifies the Java package and class to be used as compare values for the Java class filter.

Print this topic

To view or download the PDF version of the performance topic, select Performance (about 1700 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 3600 KB).

To view or download the PDF version of the Performance Tools report information, select Performance Tools reports (about 850 KB).

You can also view or download these related topics:

- Management Central (about 250 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 660 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.

You can also view or print PDFs from the “Related information” on page 133 topic.

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 **Save Target As...** if you are using Internet Explorer. Click **Save Link As...** if you are using Netscape Communicator.
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

You need Adobe Acrobat Reader to view or print these PDFs. You can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

Plan for performance

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” on page 11
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.

“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.

» **“Determine when to use simultaneous multithreading” on page 12**
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. «

“Select a performance management strategy” on page 13
Different business needs require different performance management strategies. Here are three basic business models and their suggested performance management strategies.

Set system benchmarks

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” on page 32. Depending on your environment you may need to maintain more detailed information using “Performance explorer” on page 121.

Setting up a benchmark requires:

- That the correct iSeries(TM) 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(R) 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. «

Determine when and how to expand your system

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(R) offers several tools that help you with your capacity planning, resource estimating, and sizing:

"IBM Performance Management for eServer iSeries" on page 96
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 OS/400(R) 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 for iSeries" on page 132
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(R), Java(TM), and WebSphere(R), or the consolidation of multiple AS/400(R) or iSeries traditional OS/400 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 (AS/400) - Predict" on page 133
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" on page 13 for more information about creating and implementing a performance strategy.

Determine when to use simultaneous multithreading

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. @server 5xx models support this approach through a concept called simultaneous multithreading (SMT). This approach, called hyperthreading on some Intel(R) 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(TM) 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.

Select a performance management strategy

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

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$^{(TM)}$ 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" on page 32
Collect sample data at user-defined intervals for later analysis.

"Graph history" on page 95
Display performance data collected with Collection Services.

"IBM Performance Management for eServer iSeries" on page 96
Automate the collection, archival, and analysis of system performance data.

"Performance Tools" on page 111
Gather, analyze, and maintain system performance information.

"iSeries$^{(TM)}$ Navigator monitors" on page 87
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" on page 32
Collect sample data at user-defined intervals for later analysis.

"Graph history" on page 95
Display performance data collected with Collection Services.

"IBM Performance Management for eServer iSeries" on page 96
Automate the collection, archival, and analysis of system performance data.

"Performance Tools" on page 111
Gather, analyze, and maintain system performance information.

"iSeries(TM) Navigator monitors" on page 87
Observe graphical representations of iSeries system performance, and automate responses to predefined events or conditions.

"Performance explorer" on page 121
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" on page 32
Collect sample data at user-defined intervals for later analysis.

"Graph history" on page 95
Display performance data collected with Collection Services.

"IBM Performance Management for eServer iSeries" on page 96
Automate the collection, archival, and analysis of system performance data.

"Performance Tools" on page 111
Gather, analyze, and maintain system performance information.

"iSeries(TM) Navigator monitors" on page 87
Observe graphical representations of iSeries system performance, and automate responses to predefined events or conditions.

"Performance explorer" on page 121
Collect detailed information about a specific application or system resource.

"iDoctor for iSeries" on page 128
Analyze trace data to improve system and application performance.

"Performance Trace Data Visualizer (PTDV)" on page 129
View trace data from a Java(TM) application.

Set up your environment to manage performance

The iSeries(TM) 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" on page 32**
> 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.

"IBM Performance Management for eServer iSeries" on page 96
PM iSeries uses Collection Services to gather non-proprietary performance data, and sends it to IBM(R) 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(TM) Navigator monitors" on page 87
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.

Manage iSeries(TM) performance

Successfully managing performance ensures that your system is efficiently using resources and that your server provides the best possible services to your users and to your business needs. Moreover, effective performance management can help you quickly respond to changes in your system and can save you money by postponing costly upgrades and service fees.

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.

Use the following topics to learn how to maintain your system's performance and to resolve performance problems.

"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.

"Research a performance problem" on page 16
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.

"Display performance data" on page 21
After you have collected performance data, learn how to display the data using the most appropriate tool for your purposes.

"Tune performance" on page 22
When you have identified a performance problem, you will want to tune the system to fix it.

"Manage e-business performance" on page 25
Managing performance in an e-business environment introduces several new problems for the OS/400(R) administrator. Refer to this topic for information and resources to help you plan for, track, and improve performance for your e-business applications.

Track performance

Tracking system performance for the iSeries(TM) 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.

The iSeries 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” on page 32
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” on page 95
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.

“IBM Performance Management for eServer iSeries” on page 96
PM iSeries automates the collection, archival, and analysis of system performance data and returns clear reports to help you manage system resources and capacity.

Research a 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 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 iSeries$^{(TM)}$ system, and potentially make changes to the originating program.

To learn more about collecting performance data, use the following topics to learn how and when to use some of the performance management applications.

“Identify performance problem” on page 17
Learn the common steps involved with identifying a performance problem.

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

"Collect system performance data" on page 19
Collection Services regularly collects information about system performance. Often, analyzing performance data begins with this information.

"Collect information about system resource utilization" on page 19
Several tools monitor how resources like 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 19
An application may be performing slowly for a variety of reasons. You can use several of the tools included in OS/400$^{(R)}$ and other licensed programs to help you get more information.

"Scenario: Improve system performance after an upgrade or migration" on page 21
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.

Identify 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 the OS/400$^{(R)}$ operating system 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

When performance problems occur on the iSeries$^{(TM)}$ 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.
Main storage	Investigate faulting and the wait-to-ineligible transitions.	• Work with Disk Status (WRKDSKSTS) command • 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.
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**.
» 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? «

Collect system performance data

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(TM) 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" on page 32
See how to collect performance data for analysis and how to customize your collections.

"Performance data files" on page 83
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 "What's new: Collection Services" on page 3 to start, end, and cycle collections, as well as to change and retrieve system parameters for the data collected.

Collect information about system resource utilization

Many tools are available to help you monitor and track the way the iSeries(TM) 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(TM) Navigator monitors" on page 87
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.

""Work with" commands for OS/400 performance" on page 129
OS/400 includes several important functions to help you manage and maintain system performance.

"IBM Performance Management for eServer iSeries" on page 96
PM iSeries uses Collection Services to gather non-proprietary performance data and sends it to IBM(R) 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.

Collect information about an application's performance

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 "iSeries(TM) Navigator monitors" on page 87 to track individual server performance and "Performance Tools" on page 111 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" on page 121
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 (PTDV)" on page 129
This tool is a Java(TM) application that can be used for performance analysis of applications running on iSeries. PTDV works with the performance explorer function of the OS/400(R) 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, as well as information about Java locking behavior, can be displayed. There is also support for performance explorer events generated by the WebSphere(R) 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.

"iDoctor for iSeries" on page 128
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
OS/400 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 data."

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" on page 32 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.

Scenario: Improve system performance after an upgrade or migration

You recently upgraded your iSeries(TM) 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.

Isolate changes

Several problems may result in decreased performance after upgrading the operating system. You can use the performance management tools included in OS/400(R) 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 "Performance Tools reports" on page 118 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" on page 24.
3. Run the "Performance Tools reports" on page 118 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 for iSeries" on page 128 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

For more information about resolving performance problems after a major system change, refer to the **iSeries Performance Capabilities Reference** .

Display performance data

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" on page 32 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:

""Work with" commands for OS/400 performance" on page 129
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" on page 113
The Performance Tools licensed program includes a plug-in for iSeries(TM) 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.

"iSeries(TM) Navigator monitors" on page 87
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:

"IBM Performance Management for eServer iSeries" on page 96
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" on page 95
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.

Tune performance

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. For more information about the performance monitoring steps that must precede performance tuning, see "Manage iSeries(TM) performance" on page 15.

IBM(R) 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 "Extended Adaptive Cache" on page 130.

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. See the Thread affinity system value for more information. To find out about simultaneous multithreading, see the Processor multitasking system value.

For more information about performance tuning, select from the following topics:

"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.

"Automatically tune performance" on page 24
Most users should set up the system to make performance adjustment automatically. When new systems are shipped, they are configured to adjust automatically.

Basic performance tuning

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
- Determine initial user pool sizes

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 sign-off 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, refer to "Research a performance problem" on page 16 to identify the source of the performance problem and to make specific corrections.

Automatically tune performance

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(TM) 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.

For more information about memory pools, see Memory Pools.

Manage e-business performance

Performance in an e-business environment presents several complex problems to the iSeries(TM) system 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.

The following topics can help you become familiar with some of the important considerations for maximizing your server's e-business performance, and will provide links to additional resources for detailed recommendations and examples.

"Client performance"
While the iSeries 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.

"Network performance" on page 26
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 iSeries communication resources.

"Java(TM) performance in OS/400(R)" on page 26
OS/400 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 in OS/400, and how to get the best possible performance from Java-based applications.

"IBM HTTP Server performance" on page 27
The IBM(R) HTTP server is often an important part of e-business performance on the iSeries server. IBM provides several options and configuration choices that allow you to get the most out of this server.

"WebSphere performance" on page 28
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.

In addition to these specific recommendations, administrators should also be familiar with the following topics:

- Work management
- Java for iSeries
- HTTP server
- Domino(R) for iSeries sizing and performance tuning

Client performance

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 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 iSeries$^{(TM)}$ server.

Refer to the following topics for more information:

"Collection Services" on page 32
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.

"iSeries$^{(TM)}$ Navigator monitors" on page 87
You can use the system monitors to provide information about how system resources, including communications hardware, are being used on an iSeries server. In particular, the line utilization and IOP metrics within the system monitor can provide valuable data about network performance.

"Track performance" on page 15
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.

iSeries Network.com

This web site hosts many resources for optimizing your network plan and resources. In particular, refer to the articles "Cultivate your AS/400$^{(R)}$ Networks" and "8 tools for better network performance."

Java$^{(TM)}$ performance in OS/400$^{(R)}$

Java is often the language of choice for web-based applications. However, Java applications may require some optimization, both of the OS/400 execution environment and of the Java application, to get optimal performance.

Use the following resources to learn about the Java environment in OS/400 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" on page 19
There are several tools available to help you monitor and tune an applications performance in OS/400. 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(TM) 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(R) performance in OS/400

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(R) eServer(TM) iSeries server

This Redbook provides in introduction to J2EE, and offers suggestions and examples to help you successfully implement J2EE applications on the server.

In addition to performance information, the Java topic provides resources for developing and deploying Java applications on the server.

IBM HTTP Server performance

IBM(R) HTTP server for iSeries(TM) 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" on page 32
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 the iSeries. 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 server for optimal performance. In particular, see chapter 6: Web Server and Web Commerce, for HTTP server performance specifications, planning information, and performance tips.

HTTP server (Powered by Apache)

Use this Redbook to get an in-depth description of HTTP Server (Powered by Apache) on OS/400(R), including examples for configuring HTTP Server in common usage scenarios.

AS/400(R) 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.

WebSphere performance

Managing iSeries(TM) server performance in a WebSphere(R) 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(TM) Server Pages (JSPs) and Enterprise Java Beans (EJBs).

DB2(R) 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 in OS/400(R)

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.

For other WebSphere and e-business information resources, refer to the topic WebSphere e-business administration.

Applications for performance management

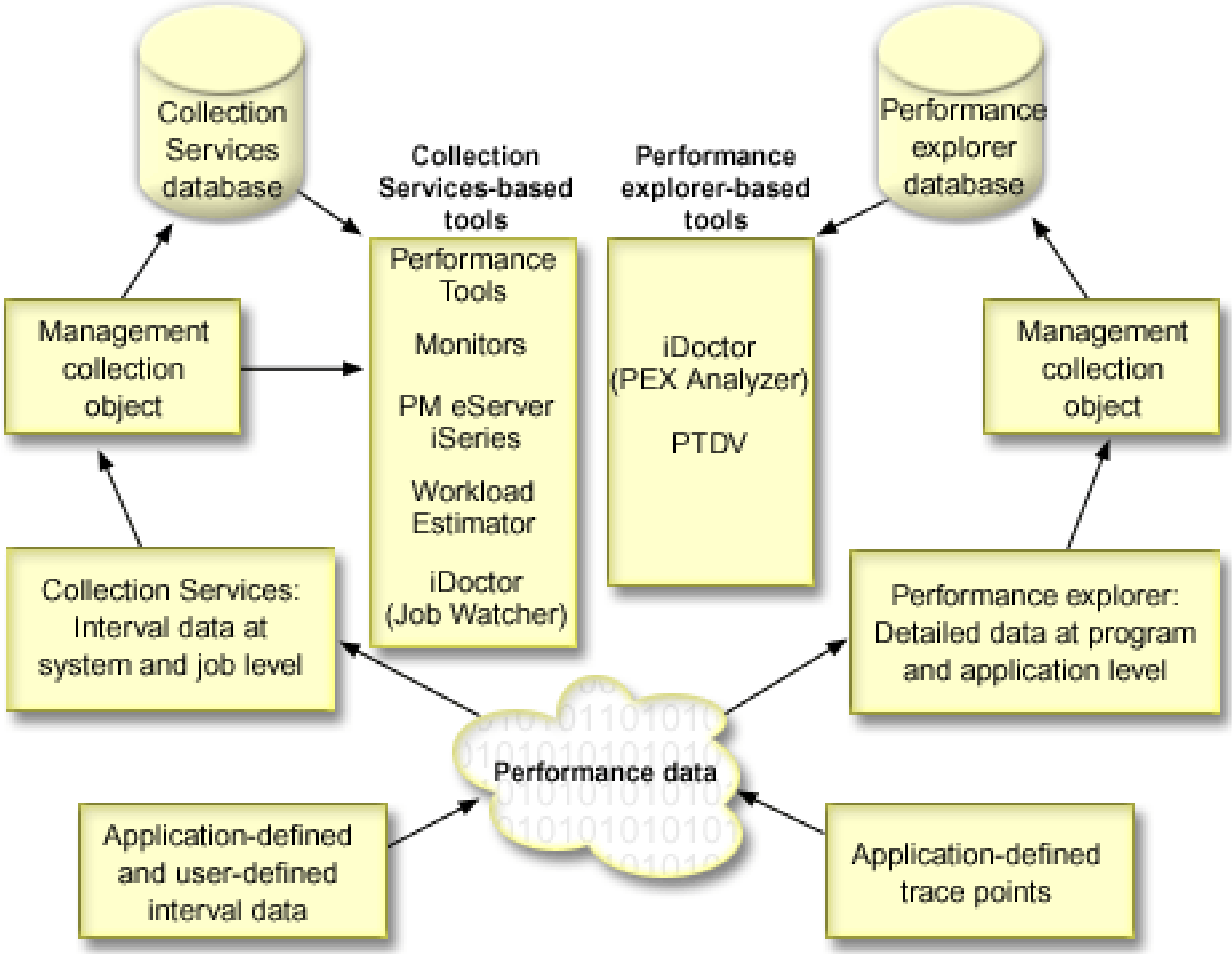
Many of the applications for performance management have several functions. Knowing exactly which component of the available suite of applications best suits a given situation can be complex. The following topics provide detailed information about each of the performance management applications, including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

As shown in the following figure, basically there are two performance collection functions on the iSeries(TM) 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(R), NetServer, or WebSphere(R).

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.


Collection Services database
Collection Services-based tools
Performance explorer-based tools
Performance explorer database
Performance Tools
Monitors
PM eServer iSeries
Workload Estimator
iDoctor (Job Watcher)
iDoctor (PEX Analyzer)
PTDV
Management collection object
Management collection object
Collection Services: Interval data at system and job level
Performance explorer: Detailed data at program and application level
Performance data
Application-defined and user-defined interval data
Application-defined trace points

"Collection Services" on page 32
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.

"Intelligent Agents" on page 65
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.

"Performance data files" on page 83
You can generate database files from the collection objects maintained by Collection Services. Use this topic to find the names, descriptions and attributes of these database files.

"iSeries$^{(TM)}$ 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.

"Graph history" on page 95
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 96
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 Tools” on page 111
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.

“Performance explorer” on page 121
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.

“iDoctor for iSeries” on page 128
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 129
Performance Trace Data Visualizer for iSeries (PTDV) is a Java$^{(TM)}$ application that can be used for performance analysis of applications running on iSeries.

“Performance Management APIs” on page 129
The 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.

““Work with” commands for OS/400 performance” on page 129
OS/400 includes several important functions to help you manage and maintain system performance.

“Extended Adaptive Cache” on page 130
Extended Adaptive Cache can improve system performance by collecting disk usage data, and then using those statistics to create a large cache, effectively reducing the physical I/O requests for the disk.

“Workload Estimator for iSeries” on page 132
Workload Estimator 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.

“iSeries$^{(TM)}$ Navigator for Wireless” on page 132
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.

“PATROL for iSeries (AS/400) - Predict” on page 133
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.

Collection Services

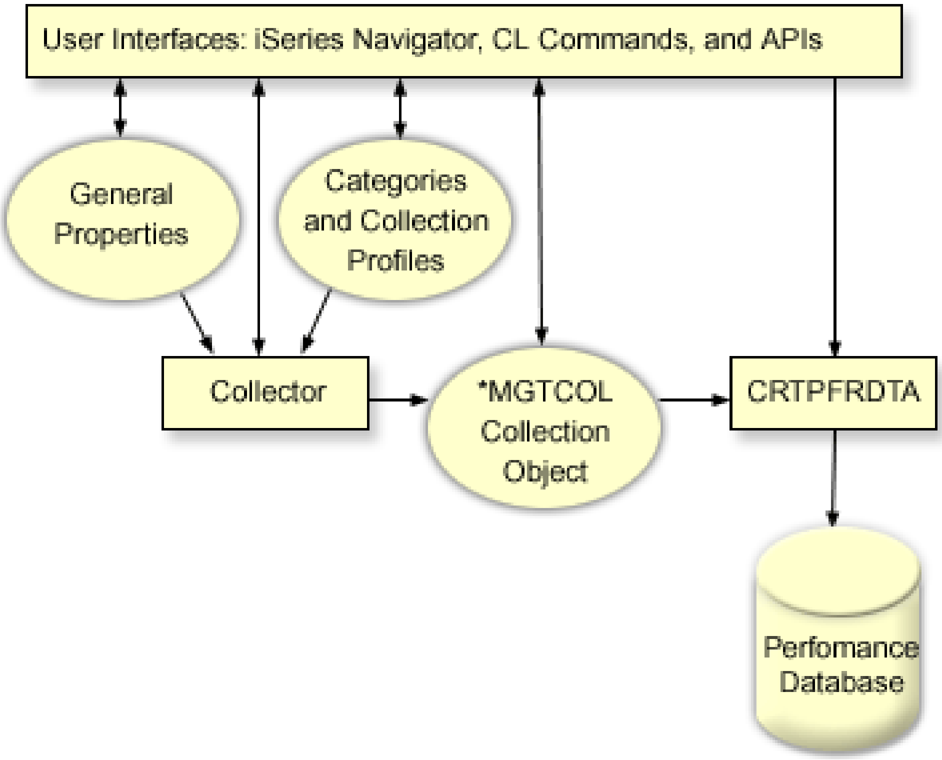
Use Collection Services to collect performance data for later analysis by the Performance Tools for iSeries(TM) licensed program or other performance report applications, iSeries Navigator monitors, and the graph history function. (If you prefer viewing real-time performance data, system monitors provide 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 OS/400(R) 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 OS/400 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 OS/400 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.


User Interfaces: iSeries Navigator, CL Commands, and APIs
General Properties
Categories and Collection Profiles
Collector
*MGTCOL Collection Object
CRTPFRDTA
Perfomance Database

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

Performance database
The database files store the data that is processed by the CRTPFRDTA command. The files can be divided into these categories: Performance data files that contain time interval data, configuration data files, 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" on page 35

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)	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 .
"Activate PM iSeries" on page 99	PM iSeries automates the start of Collection Services and then creates the database files during collection.

Collection Services tasks

You can use Collection Services and iSeries Navigator to perform a variety of data collection tasks as shown in the following table.

Task	Description
Start Collection Services in a variety of ways	Create a customized performance data collection on an individual system or groups of systems with specific performance metrics. You can also use a Start Collector API in your startup program to start performance data collections automatically. For more information about how to perform these tasks, refer to the online help. Detailed task help is available from the iSeries Navigator window. Just click **Help** from the menu bar and select **Help Topics**. Select **What can I do with . . .?** to find out what you can do and where you need to be in the iSeries Navigator window to make it happen.

Task	Description
“Create database files from Collection Services data” on page 38	Learn how to control what data gets collected as you create database files. For example, you can use Collection Services to automate the creation of performance database files or you can create database files from the collection object, where the data is stored after it has been collected. In addition, you can use these database files with PM iSeries or with the Performance Tools licensed program, or you can create your own queries to run against these files. See “Performance data files” on page 83 to find out what database files are available to you, as well as what field-level data is included in each file.
“Customize data collections” on page 40	Customize your data collections. Find information about controlling what performance data you collect and how often that data gets collected. You can also find information about important “Time zone considerations for Collection Services” on page 41.
“User-defined categories in Collection Services” on page 41	You can collect performance data from user applications by writing an exit program and integrating it into Collection Services. You can then collect this data during routine collection intervals and store it in collection objects.
“Manage collection objects” on page 50	Find the information you need to manage collection objects, including the contents of collection objects, how long collection objects are saved, and what you can do with collection objects.
“Collect information about an application’s performance” on page 19	Collection Services collects sample data. However, it does not collect trace data. Refer to this topic to learn how to collect trace data.
“User-defined transactions” on page 51	Collection Services provides APIs that allow you to define your own transactions.
» “Collecting performance data across partitions” on page 59	Collection Services can collect performance data from your iSeries partitions, regardless of the operating system running on the partition. PM iSeries will then aggregate the data before it is sent to IBM$^{(R)}$ for analysis.
“Find wait statistics for a job, task, or thread” on page 64	Identify the cause and duration of wait time experienced by a job, task or thread
“Understanding disk consumption by Collection Services” on page 64	The amount of disk space consumed by Collection Services varies significantly depending on collection interval and retention period you have selected. Refer to this topic to help you plan for Collection Services disk consumption. «

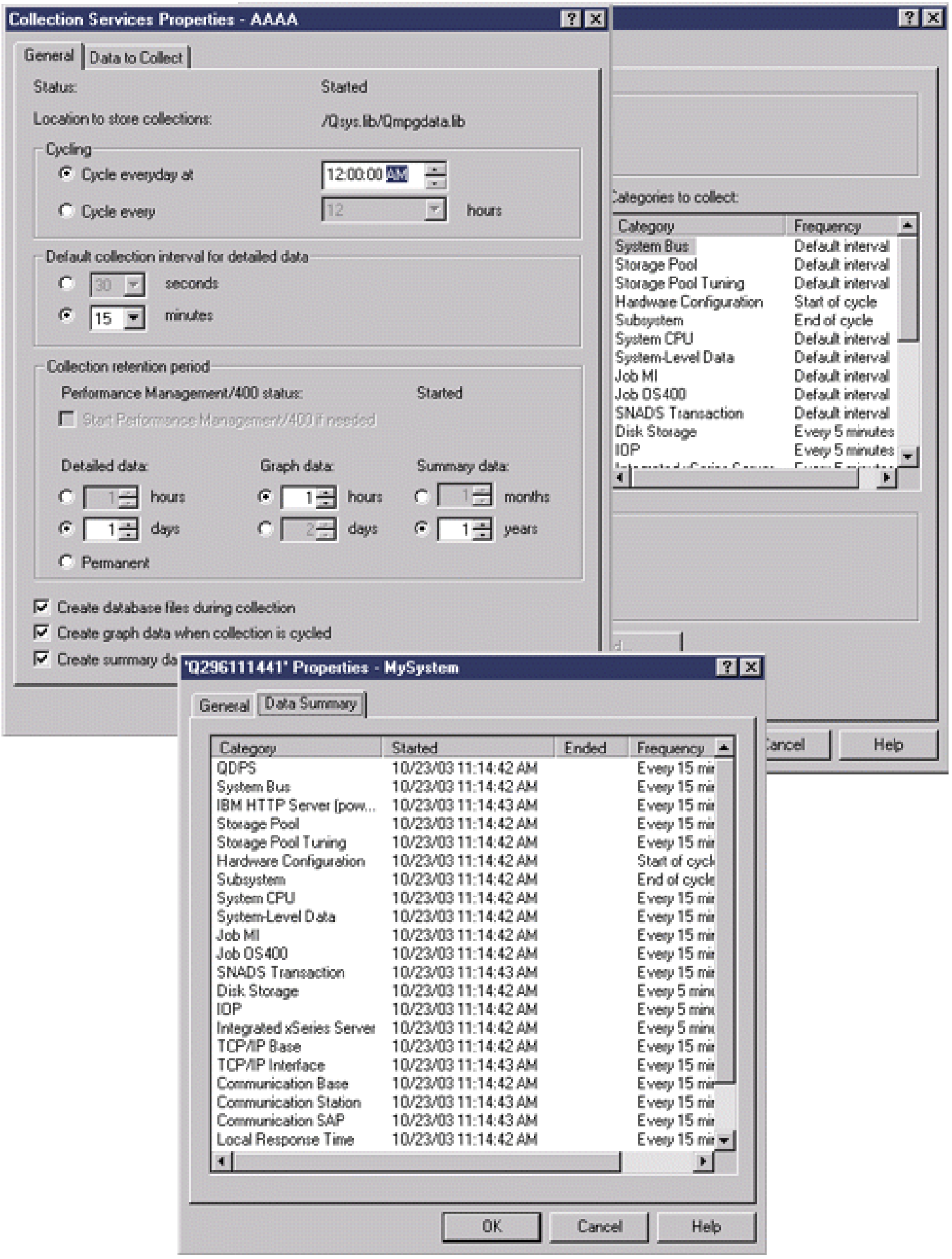
System and job monitor interaction with Collection Services

» 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:



Collection Services Properties - AAAA

Status: Started

Location to store collections: /Qsys.lib/Qmpgdata.lib

Cycling

● Cycle everyday at 12:00:00 AM

○ Cycle every 12 hours

Default collection interval for detailed data

○ 30 seconds

● 15 minutes

Collection retention period

Performance Management/400 status: Started

□ Start Performance Management/400 if needed

Detailed data: ○ 1 hours ● 1 days ○ Permanent

Graph data: ● 1 hours ○ 2 days

Summary data: ○ 1 months ● 1 years

☑ Create database files during collection

☑ Create graph data when collection is cycled

☑ Create summary da

Categories to collect:

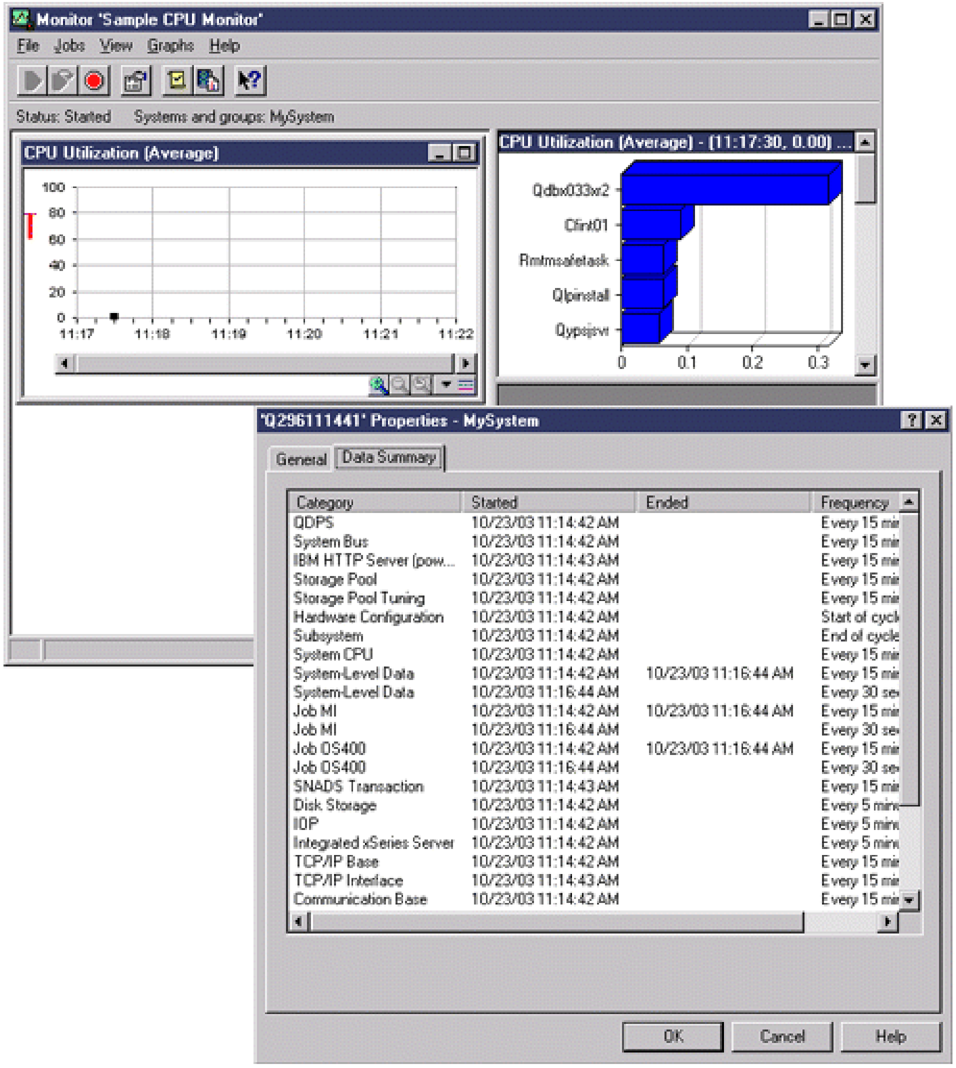
Category	Frequency
System Bus	Default interval
Storage Pool	Default interval
Storage Pool Tuning	Default interval
Hardware Configuration	Start of cycle
Subsystem	End of cycle
System CPU	Default interval
System-Level Data	Default interval
Job MI	Default interval
Job OS400	Default interval
SNADS Transaction	Default interval
Disk Storage	Every 5 minutes
IOP	Every 5 minutes

'Q296111441' Properties - MySystem

Category	Started	Ended	Frequency
QDPS	10/23/03 11:14:42 AM		Every 15 mi
System Bus	10/23/03 11:14:42 AM		Every 15 mi
IBM HTTP Server (pow...	10/23/03 11:14:43 AM		Every 15 mi
Storage Pool	10/23/03 11:14:42 AM		Every 15 mi
Storage Pool Tuning	10/23/03 11:14:42 AM		Every 15 mi
Hardware Configuration	10/23/03 11:14:42 AM		Start of cycl
Subsystem	10/23/03 11:14:42 AM		End of cycle
System CPU	10/23/03 11:14:42 AM		Every 15 mi
System-Level Data	10/23/03 11:14:42 AM		Every 15 mi
Job MI	10/23/03 11:14:42 AM		Every 15 mi
Job OS400	10/23/03 11:14:42 AM		Every 15 mi
SNADS Transaction	10/23/03 11:14:43 AM		Every 15 mi
Disk Storage	10/23/03 11:14:42 AM		Every 5 min
IOP	10/23/03 11:14:42 AM		Every 5 min
Integrated xSeries Server	10/23/03 11:14:42 AM		Every 5 min
TCP/IP Base	10/23/03 11:14:42 AM		Every 15 mi
TCP/IP Interface	10/23/03 11:14:43 AM		Every 15 mi
Communication Base	10/23/03 11:14:42 AM		Every 15 mi
Communication Station	10/23/03 11:14:43 AM		Every 15 mi
Communication SAP	10/23/03 11:14:43 AM		Every 15 mi
Local Response Time	10/23/03 11:14:42 AM		Every 15 mi

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.


Monitor 'Sample CPU Monitor'
File Jobs View Graphs Help
CPU Utilization (Average)
100
80
60
40
20
0
11:17
11:18
11:19
11:20
11:21
11:22
CPU Utilization (Average) - (11:17:30, 0.00) ...
Qdbx033xr2
Cfint01
Rmtmsafetask
Qlpinstall
Qypsjsvr
0
0.1
0.2
0.3
'Q296111441' Properties - MySystem
General
Data Summary
OK
Cancel
Help

Status: Started Systems and groups: MySystem

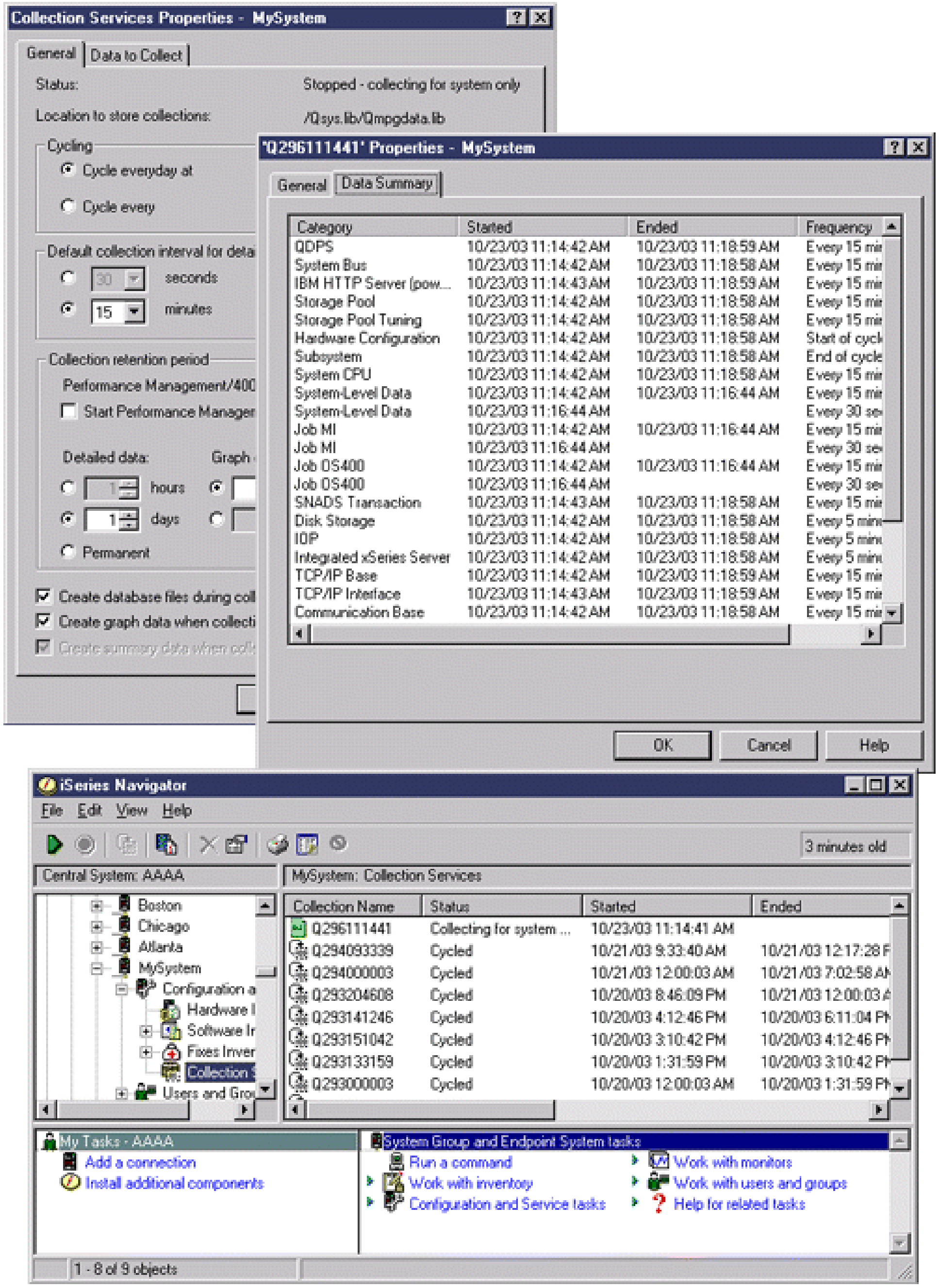
Category	Started	Ended	Frequency
QDPS	10/23/03 11:14:42 AM		Every 15 mi
System Bus	10/23/03 11:14:42 AM		Every 15 mi
IBM HTTP Server (pow...	10/23/03 11:14:43 AM		Every 15 mi
Storage Pool	10/23/03 11:14:42 AM		Every 15 mi
Storage Pool Tuning	10/23/03 11:14:42 AM		Every 15 mi
Hardware Configuration	10/23/03 11:14:42 AM		Start of cycl
Subsystem	10/23/03 11:14:42 AM		End of cycle
System CPU	10/23/03 11:14:42 AM		Every 15 mi
System-Level Data	10/23/03 11:14:42 AM	10/23/03 11:16:44 AM	Every 15 mi
System-Level Data	10/23/03 11:16:44 AM		Every 30 se
Job MI	10/23/03 11:14:42 AM	10/23/03 11:16:44 AM	Every 15 mi
Job MI	10/23/03 11:16:44 AM		Every 30 se
Job OS400	10/23/03 11:14:42 AM	10/23/03 11:16:44 AM	Every 15 mi
Job OS400	10/23/03 11:16:44 AM		Every 30 se
SNADS Transaction	10/23/03 11:14:43 AM		Every 15 mi
Disk Storage	10/23/03 11:14:42 AM		Every 5 min
IOP	10/23/03 11:14:42 AM		Every 5 min
Integrated xSeries Server	10/23/03 11:14:42 AM		Every 5 min
TCP/IP Base	10/23/03 11:14:42 AM		Every 15 mi
TCP/IP Interface	10/23/03 11:14:43 AM		Every 15 mi
Communication Base	10/23/03 11:14:42 AM		Every 15 mi

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.



Status: Stopped - collecting for system only

Location to store collections: /Qsys.lib/Qmpgdata.lib

Cycling

● Cycle everyday at

○ Cycle every

Default collection interval for detai

○ 30 seconds

● 15 minutes

Collection retention period

Performance Management/400

□ Start Performance Manager

Detailed data:

○ 1 hours

● 1 days

○ Permanent

Graph

☑ Create database files during coll

☑ Create graph data when collecti

☑ Create summary data when coll

Category	Started	Ended	Frequency
QDPS	10/23/03 11:14:42 AM	10/23/03 11:18:59 AM	Every 15 mi
System Bus	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 15 mi
IBM HTTP Server (pow...	10/23/03 11:14:43 AM	10/23/03 11:18:59 AM	Every 15 mi
Storage Pool	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 15 mi
Storage Pool Tuning	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 15 mi
Hardware Configuration	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Start of cycl
Subsystem	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	End of cycle
System CPU	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 15 mi
System-Level Data	10/23/03 11:14:42 AM	10/23/03 11:16:44 AM	Every 15 mi
System-Level Data	10/23/03 11:16:44 AM		Every 30 se
Job MI	10/23/03 11:14:42 AM	10/23/03 11:16:44 AM	Every 15 mi
Job MI	10/23/03 11:16:44 AM		Every 30 se
Job OS400	10/23/03 11:14:42 AM	10/23/03 11:16:44 AM	Every 15 mi
Job OS400	10/23/03 11:16:44 AM		Every 30 se
SNADS Transaction	10/23/03 11:14:43 AM	10/23/03 11:18:58 AM	Every 15 mi
Disk Storage	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 5 min
IOP	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 5 min
Integrated xSeries Server	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 5 min
TCP/IP Base	10/23/03 11:14:42 AM	10/23/03 11:18:59 AM	Every 15 mi
TCP/IP Interface	10/23/03 11:14:43 AM	10/23/03 11:18:59 AM	Every 15 mi
Communication Base	10/23/03 11:14:42 AM	10/23/03 11:18:58 AM	Every 15 mi

Collection Name	Status	Started	Ended
Q296111441	Collecting for system ...	10/23/03 11:14:41 AM	
Q294093339	Cycled	10/21/03 9:33:40 AM	10/21/03 12:17:28 F
Q294000003	Cycled	10/21/03 12:00:03 AM	10/21/03 7:02:58 AI
Q293204608	Cycled	10/20/03 8:46:09 PM	10/21/03 12:00:03 A
Q293141246	Cycled	10/20/03 4:12:46 PM	10/20/03 6:11:04 PI
Q293151042	Cycled	10/20/03 3:10:42 PM	10/20/03 4:12:46 PI
Q293133159	Cycled	10/20/03 1:31:59 PM	10/20/03 3:10:42 PI
Q293000003	Cycled	10/20/03 12:00:03 AM	10/20/03 1:31:59 PI

1 - 8 of 9 objects

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 database files from an existing collection object" on page 40 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$^{(TM)}$ Navigator interface or the CRTPFRDTA command.

- You can activate PM iSeries, 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 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. For more information about Performance Tools, see the Performance Tools book .

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**.

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” on page 111 or other applications to produce performance reports.

Customize data collections

When you use Collection Services to “Collection Services” on page 32, 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 OS/400(R) 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(TM) 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” on page 95, 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.

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.

For more information about using Collection Services to collect performance data, see “Collection Services” on page 32.

User-defined categories in Collection Services

The user-defined categories function in “Collection Services” on page 32 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. Refer to “Collection program recommendations and requirements” for more information.
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. See the API descriptions for more information:
   - 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. See the API descriptions for more information:
   - Retrieve active management collection object name: QpmRtvActiveMgtcolName (Used only for querying the collection object in real-time.)
   - Retrieve management collection object attributes: QpmRtvMgtcolAttrs
   - Open management collection object: QpmOpenMgtcol
   - Close management collection object: QpmCloseMgtcol
   - Open management collection object repository: QpmOpenMgtcolRepo
   - Close management collection object repository: QpmCloseMgtcolRepo
   - Read management collection object data: QpmReadMgtcolData

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

You can also implement the Java(TM) 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, refer to the Javadocs in the Information Center, or download the javadocs in a .zip file.

For an example implementation, see “Example: Implementing user-defined categories” on page 44

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 de-register the data queue, refer to the following API descriptions:

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

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 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.

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(TM) 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.

Record	Description
Stop	Collection Services automatically creates this record to indicate the end of a data collection session. If the collections for the user-defined category were restarted without ending or cycling Collection Services, you can optionally include a control record followed by additional interval records after the stop record.

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

- "Example: data collection program"
- "Example: Program to register the data collection program" on page 47
- "Example: Program to query the collection object" on page 48

Code example disclaimer

IBM$^{(R)}$ 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. 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 INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Example: data collection program: The following program collects some test data and stores it in a data buffer, which Collection Services copy to the collection object. For more information about the collection program, refer to "Collection program recommendations and requirements" on page 42

Note: Read the "Code disclaimer information" on page 140 for important legal information.

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() */
```

Example: Program to register the data collection program: The following 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: Read the “Code disclaimer information” on page 140 for important legal information.

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,
                "1234567890123456789012345678901234567890123456789O", 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: The following sample program illustrates how to query the data stored in the collection object using the Java$^{(TM)}$ classes shipped in the ColSrv.jar file in QIBM/ProdData/OS400/CollectionServices/lib.

Note: Read the "Code disclaimer information" on page 140 for important legal information.

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 “Collection Services” on page 32, 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$^{(TM)}$ 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. 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 from Collection Services data" on page 38 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" on page 95 to graphically view the data in the management collection object.

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
- End transaction: QYPEENDT, qypeEndTransaction
- Log transaction: QYPELOGT, qypeLogTransaction (Used only by performance explorer)
- Add trace point: QYPEADDT, qypeAddTracePoint (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 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. For more information, refer to the QAPMUSRTNS file description and the API descriptions.

For a sample implementation, refer to the examples in “C++ example: Integrating user-defined transactions into Collection Services” or “Java(TM) example: Integrating user-defined transactions into Collection Services” on page 56(TM).

Note: Read the “Code disclaimer information” on page 140 for important legal information.

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” on page 121 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.

For a detailed description, refer to the API descriptions and the usage notes in the Start transaction API description.

C++ example: Integrating user-defined transactions into Collection Services: The following 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: Read the “Code disclaimer information” on page 140 for important legal information.

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

#define _MULTI_THREADED

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

// Constants
#define maxThreads 10000

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

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


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

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

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

  struct timespec ts = {0, 0};
  error_code_t errCode;

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

  unsigned int typCnt;
  unsigned int tnsCnt;
  int rc;


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

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

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

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

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

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

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

  return NULL;
}


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

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

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

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


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

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

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

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

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

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

}  /* end of Main */
```

Java$^{(TM)}$ example: Integrating user-defined transactions into Collection Services: The following 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: Read the "Code disclaimer information" on page 140 for important legal information.

```
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 */
```

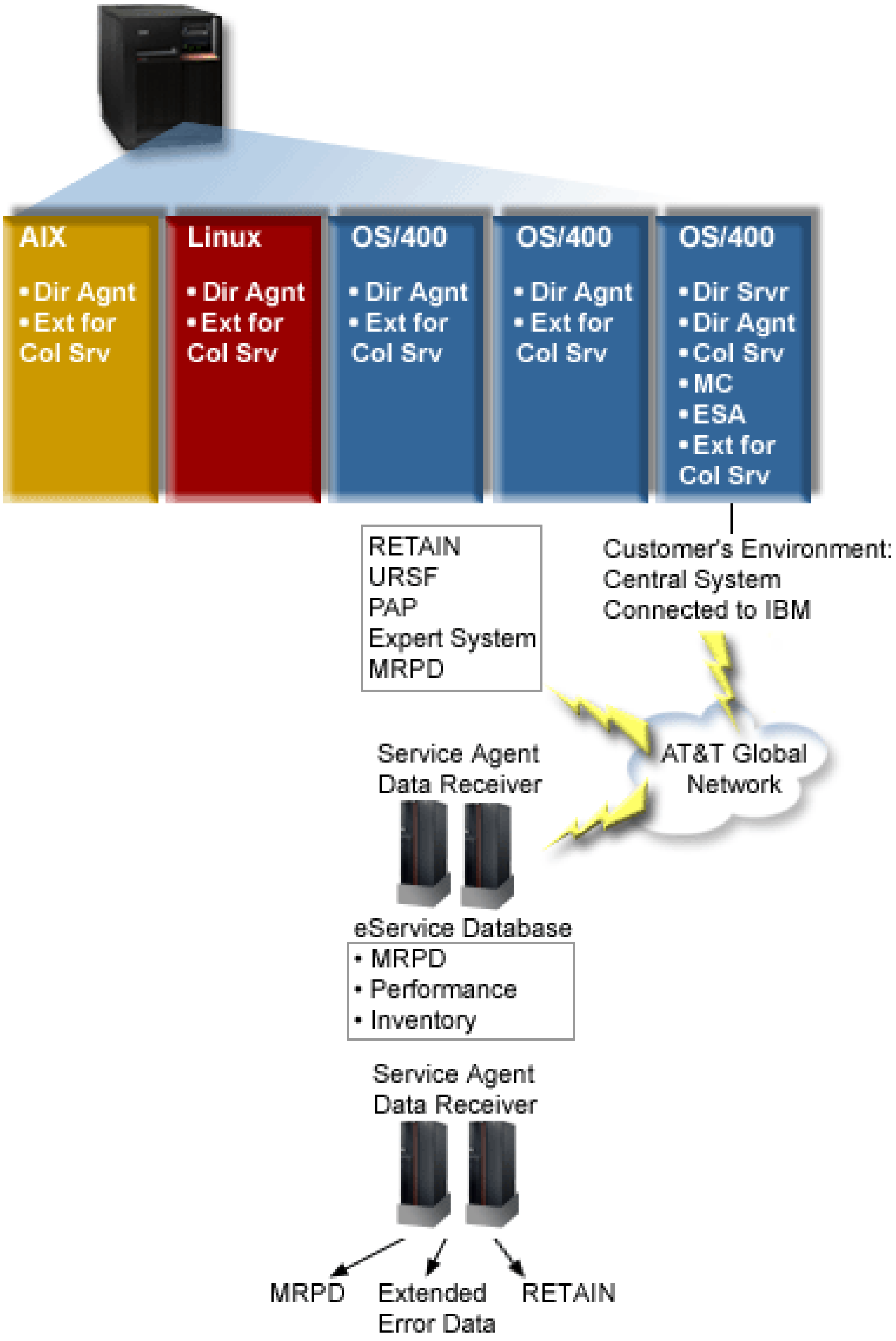
Collecting performance data across partitions

» IBM(R) Performance Management for @server iSeries(TM) (PM eServer(TM) iSeries or PM 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 very straightforward process. However, if your system has been partitioned into multiple OS/400(R) 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(R) or Linux(TM), or if any of the OS/400(R) partitions have PM iSeries(R) 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 iSeries 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 will help 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 OS/400 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(TM) (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.


AIX
• Dir Agnt
• Ext for Col Srv
Linux
• Dir Agnt
• Ext for Col Srv
OS/400
• Dir Agnt
• Ext for Col Srv
OS/400
• Dir Agnt
• Ext for Col Srv
OS/400
• Dir Srvr
• Dir Agnt
• Col Srv
• MC
• ESA
• Ext for Col Srv
RETAIN
URSF
PAP
Expert System
MRPD
Customer's Environment: Central System Connected to IBM
AT&T Global Network
Service Agent Data Receiver
eService Database
• MRPD
• Performance
• Inventory
Service Agent Data Receiver
MRPD
Extended Error Data
RETAIN

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:
   - OS/400, version 5 release 3
   - AIX 5L(TM), version 5.3
   - Red Hat Enterprise Linux(TM) AS, version 3.0, for IBM PowerPC(R)
   - SUSE LINUX(TM) Enterprise Server 8 for IBM pSeries(R) and IBM iSeries
   - SUSE LINUX(TM) Enterprise Server 9 for IBM pSeries(R) and IBM iSeries
3. Ensure that you have applied the following Collection Services PTF fixes to the partition that will act 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(TM) to install IBM Director Server on the OS/400 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 OS/400 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 OS/400 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 OS/400 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 OS/400 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 OS/400 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 OS/400 partition that you want to manage.

   b. Distribute ColSrvLparDataSubagt.TWGSubagent to the Director Multiplatform extensions directory, /qibm/userdata/director/classes/extensions, on the OS/400 partition that you want to manage.

   c. Distribute ColSrvDir.jar to Director Multiplatform classes directory, /qibm/userdata/director/classes, on the OS/400 partition that you want to manage.

10. On each Linux(TM) partition, install the Director Multiplatform extension for Collection Services by installing the Collection Services RPM file, ColSrvDirExt-5.3.0-1.ppc64.rpm.

    a. Distribute the Collection Services RPM file from the management partition to the Linux(TM) 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 OS/400 management partition directory, /qibm/proddata/os400/collectionservices/lib/ColSrvDirExt-5.3.0-1.ppc64.rpm.

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

```
rpm -Uhv --force ColSrvDirExt-5.3.0-1.ppc64.rpm
```

11. On each AIX(R) partition, install the Director Multiplatform extension for Collection Services by installing the Collection Services package, aix-ColSrvDirExt-5.3.bff.

    a. Distribute the Collection Services package file from the management server to the AIX(R) 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 OS/400 management partition directory, /qibm/proddata/os400/collectionservices/lib/aix-ColSrvDirExt-5.3.bff.

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

```
installp -Fac -d  aix-ColSrvDirExt-5.3.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” on page 99, 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)
```

Related information

- IBM Director Multiplatform
- "IBM Performance Management for eServer iSeries" on page 96
- IBM Virtualization Engine
- Management Central
- Partitioning the server
- "Send PM iSeries(TM) data with Service Agent over Extreme Support (Universal Connection)" on page 100

«

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.

For more information about using and accessing this information, refer to the Performance database files QAPMJOBWT and QAPMJOBWTD.

Note: To query these files, your system CCSID ID must be set to your primary language (as opposed to 65535 -binary data). «

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.

«

Intelligent Agents

Intelligent agents are Java(TM)-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$^{(TM)}$ 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.

"Develop Agents" on page 68
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.

"Set up your agent environment" on page 70
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.

"Manage agents" on page 79
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.

Intelligent Agent concepts

Concepts:

"ABLE agents"
The Intelligent Agents console for iSeries$^{(TM)}$ Navigator works with ABLE (Agent Building and Learning Environment) agents. ABLE is a Java$^{(TM)}$ framework and a toolkit used for building multiagent intelligent autonomic systems.

"Agent platform" on page 67
Agent Services live on your system or across your distributed platform, and are responsible for the life cycle, security, and behavior of your agent.

ABLE agents: The Intelligent Agents console for iSeries$^{(TM)}$ Navigator works with Agent Building and Learning Environment (ABLE) agents. ABLE agents are Java$^{(TM)}$ 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:

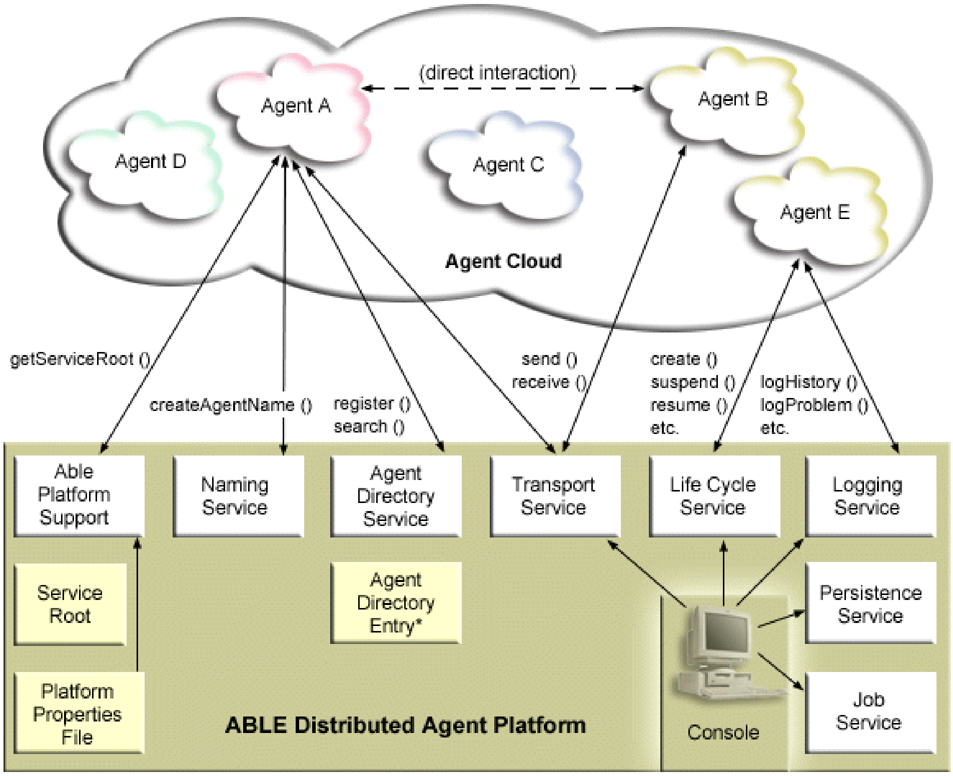
- Learn from experience and predict future states
- Analyze metric data using classification and clustering algorithms to detect complex states and diagnose problems
- Interface with other autonomic components via web services
- Reason using domain-specific Java application objects
- Use powerful machine 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**. For more information on developing ABLE agents, and obtaining the ABLE toolkit and its corresponding documentation, see: "Develop Agents" on page 68

Agent platform: The Intelligent Agents console in iSeries(TM) 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(TM) 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(TM) 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 .

Develop Agents

You can use the Agent Building and Learning Environment (ABLE) toolkit to develop your own hybrid intelligent agents. This Java(TM) 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(TM) 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**.

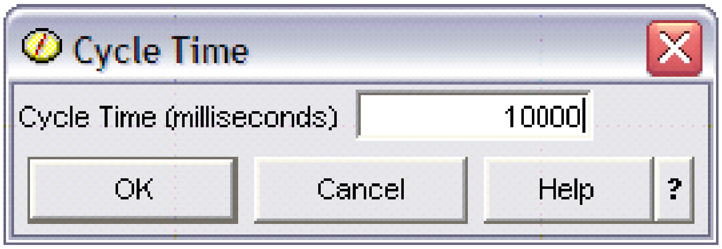
AbleEserverTemplateAgent makes use of many of the features available when developing agents using the ABLE framework. It demonstrates how an agent would create a set of capabilities that could be managed through the console. It includes a **Customize** panel that can be used to alter agent settings and 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.

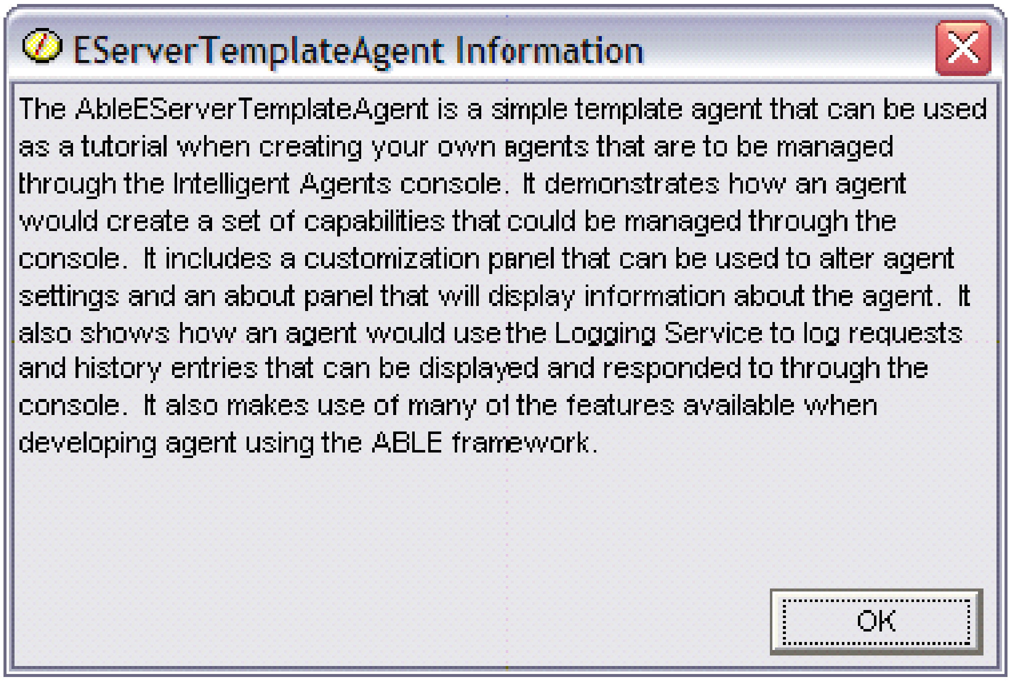


Cycle Time

Cycle Time (milliseconds) 10000

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.



EServerTemplateAgent 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 ot the features available when developing agent using the ABLE framework.

Figure 2: Viewing the template agent's about panel

Set up your agent environment

The iSeries(TM) Navigator Intelligent Agents console functions by connecting to an agent platform running on your system, or across a distributed network. The "Agent platform" on page 67 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:

> **"Configure your agent platform"**
> Before you begin using the Intelligent Agents console in iSeries Navigator, you first need to configure the agent platform.
>
> **"Secure your agent environment" on page 73**
> 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.
>
> **"Start the agent platform" on page 78**
> After you define the agent platform and optionally secure your platform, you will need to start all the Java(TM) Virtual Machines associated with your agent services using iSeries CL commands.

Configure your agent platform: This topic provides a brief overview of the agent platform, and then provides detailed configuration steps for modifying the platform preferences file.

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(TM) 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" on page 67.

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" on page 78 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" on page 73 topic. The following is an example of how a set of agent pools could be defined:

   ```
   #----------------------------------------------------------------------
   # 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(R) 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” on page 78

Secure your agent environment: 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.

Before you turn on security, you need to configure ensure the following steps have been performed:

1. **“Configure your platform to use Kerberos”**
   The intelligent agent platform uses Kerberos principals to authenticate users and services on or across the agent platform. Authentication of principals is completed through a centralized server called a key distribution center (KDC), and in V5R3, a native Kerberos KDC is provided on iSeries$^{(TM)}$.
2. **“Configure platform security” on page 76**
   When security is turned on, **ableplatform.preferences** acts as a policy file for the security of the platform it defines. The following topic provides steps for configuring principals, trustlevels, and permissions.

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(TM) 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 0S/400(R), a KDC running on any platform will work. If you cannot find an existing KDC to use, you can create your own. In V5R3, OS/400 supports a native Kerberos server in OS/400 PASE. You can configure and manage a Kerberos server from your iSeries(TM) system. To configure a Kerberos server in OS/400 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 OS/400 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 OS/400 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.

   For more information on how to configure a KDC on iSeries, see Configure a Kerberos server in OS/400 PASE.
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:

```
[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 iSeries 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
```

  For more detailed instructions on how to point your iSeries to the KDC you have created, see Configure network authentication.

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 iSeries:

   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:
        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 on iSeries, 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, see:
Add OS/400 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 Window s 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 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 iSeries:
   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 iSeries 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 platform security."

Configure platform security: Before you begin, ensure that you have "Configure your platform to use Kerberos" on page 73.

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, trustlevels, 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):

```
#-----------------------------------------------------------------------
# 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 trustlevel 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 (Java$^{(TM)}$ Virtual Machines) 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 “Start the agent platform” for security changes to take effect.

Start the agent platform: After you have “Configure your agent platform” on page 70 and optionally configured “Secure your agent environment” on page 73, you need to start the agent platform. Because the platform is made up of one or more Java(TM) 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(TM):

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

- **STRAGTSRV** (Start Agent Services)
  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).

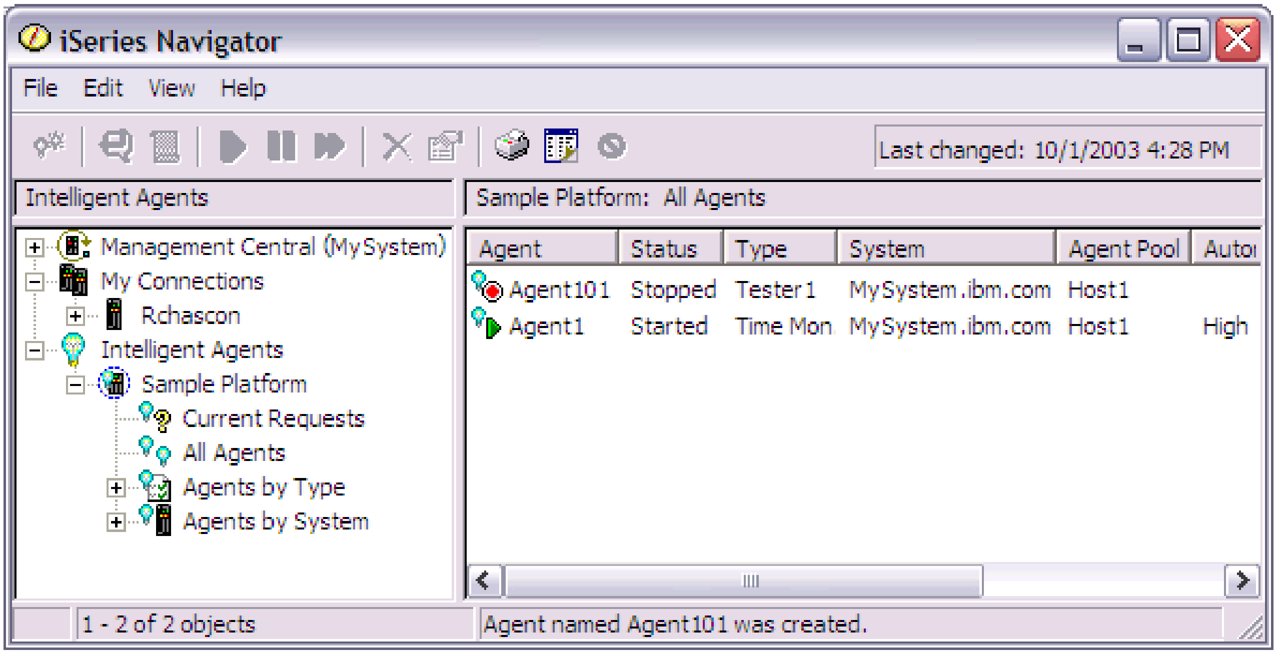
- **ENDAGTSRV** (End Agent Services)
  Ends all of the platform JVMs on this system that are specified as agent pools in the file **ableplatform.preferences**. This command starts a QAHAPLTEND job that will find and end all agent pools. ENDAGTSRV has the following keywords:

  > **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.

Manage agents

The Intelligent Agents console for iSeries(TM) 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.



Last changed: 10/1/2003 4:28 PM

Sample Platform: All Agents

Agent	Status	Type	System	Agent Pool	Autoı
Agent101	Stopped	Tester1	MySystem.ibm.com	Host1	
Agent1	Started	Time Mon	MySystem.ibm.com	Host1	High

1 - 2 of 2 objects

Agent named Agent101 was created.

Figure 1: 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.

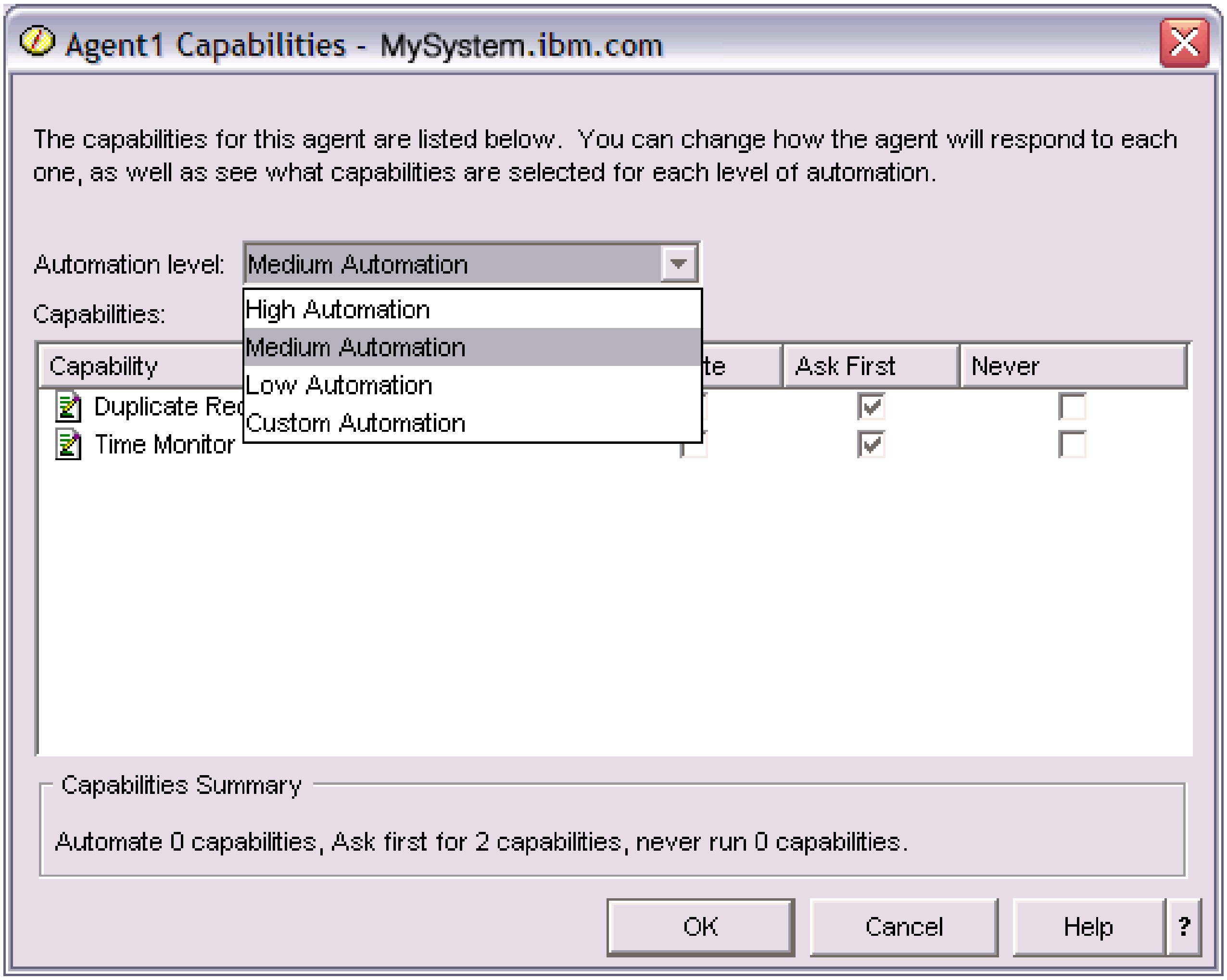
“Agent Communication” on page 82
Easily track and respond to agents that are requesting confirmation or permission to take action.

“Agent history” on page 82
The agent console logs a history of all your agents actions.

Agent Automation: 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$^{(TM)}$ 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.



Agent1 Capabilities - MySystem.ibm.com

The capabilities for this agent are listed below. You can change how the agent will respond to each one, as well as see what capabilities are selected for each level of automation.

Automation level: Medium Automation

Capabilities:

Capability	te	Ask First	Never
Duplicate Rec	☐	☑	☐
Time Monitor	☐	☑	☐

Capabilities Summary

Automate 0 capabilities, Ask first for 2 capabilities, never run 0 capabilities.

Figure 1: 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:

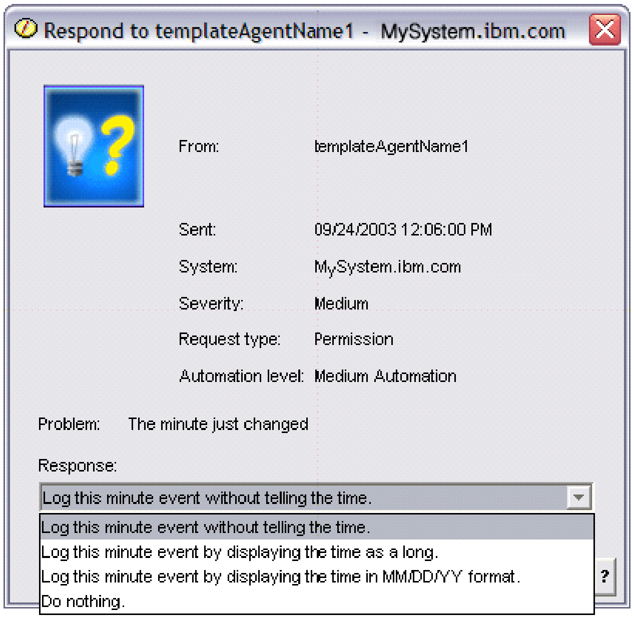
- **High automation**
  The agent will perform most actions automatically, but will ask before performing certain destructive actions. Depending on the agent, certain actions may require that the agent always request outside intervention before it performs that action, even when set to **High automation**.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


Respond to templateAgentName1 - MySystem.ibm.com

From: templateAgentName1

Sent: 09/24/2003 12:06:00 PM

System: MySystem.ibm.com

Severity: Medium

Request type: Permission

Automation level: Medium Automation

Problem: The minute just changed

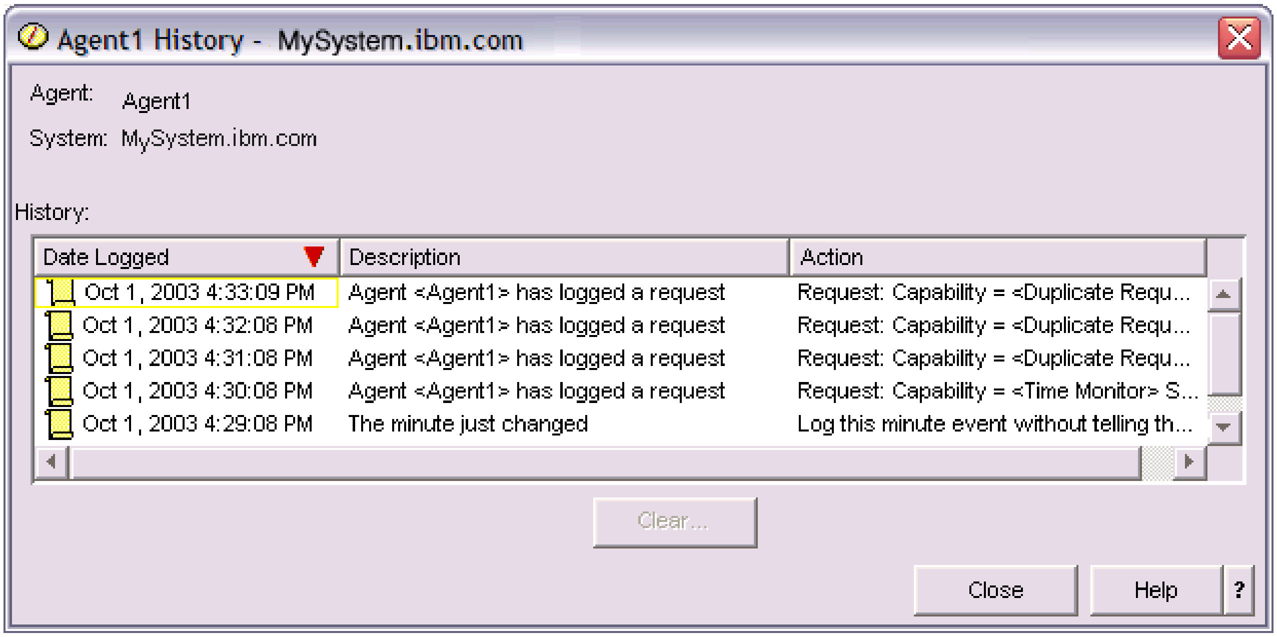
Response: Log this minute event without telling the time.

Log this minute event without telling the time.
Log this minute event by displaying the time as a long.
Log this minute event by displaying the time in MM/DD/YY format.
Do nothing.
?

Figure 1: 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 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.



Agent1 History - MySystem.ibm.com

Agent: Agent1

System: MySystem.ibm.com

History:

Date Logged	Description	Action
Oct 1, 2003 4:33:09 PM	Agent <Agent1> has logged a request	Request: Capability = <Duplicate Requ...
Oct 1, 2003 4:32:08 PM	Agent <Agent1> has logged a request	Request: Capability = <Duplicate Requ...
Oct 1, 2003 4:31:08 PM	Agent <Agent1> has logged a request	Request: Capability = <Duplicate Requ...
Oct 1, 2003 4:30:08 PM	Agent <Agent1> has logged a request	Request: Capability = <Time Monitor> S...
Oct 1, 2003 4:29:08 PM	The minute just changed	Log this minute event without telling th...

Figure 1: Viewing the history of an agent's requests and actions

To view an agent's history in iSeries$^{(TM)}$ 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**.

Performance data files

Performance data is a set of information about the operation of a system (or network of systems) that can be used to understand response time and throughput. You can use performance data to make adjustments to programs, system attributes, and operations. These adjustments can improve response times and throughputs. Adjustments can also help you to predict the effects of certain changes to the system, operation, or program.

"Collection Services" on page 32 collects performance data into a management collection object (*MGTCOL). The Create Performance Data (CRTPFRDTA) command processes data from that collection object and stores the result into performance database files. The database files are divided into the following categories:

Performance data files containing time interval data
These files contain performance data that is collected each interval. See "Performance data files containing time interval data" on page 84 data for a list of these files, with a brief description and a link to complete information about each file. To understand where the data in these files comes from, refer to "Performance data files: Collection Services system category and file relationships" on page 86. When viewing these files, you might also find the "Performance data files: File abbreviations" on page 85 useful.

Configuration data files
Configuration data is collected once per session. To understand where the data in these files comes from, refer to "Performance data files: Collection Services system category and file relationships" on page 86. You can find the QAPMCONF, QAPMHDWR, and QAPMSBSD files in the configuration data files.

Trace data files
Trace data is collected only when you choose to do so. You can find the QAPMDMPT file in the trace data files.

Additional field information such as number of bytes and buffer position is available by using the Display File Field Description (DSPFFD) command available on the system. For example, type the following on any command line:

```
DSPFFD file(QSYS/QAPMCONF)
```

For more information about iSeries(TM) performance, see "Performance," on page 1

Performance data files containing time interval data

To view complete information about a performance data file, select the file you want to view from the list below (shown in alphabetical order).

File	Description
QAPMAPPN	APPN data
QAPMASYN	Asynchronous statistics (one per link)
QAPMBSC	Binary synchronous statistics (one per link)
QAPMBUS	Bus counters (one per bus)
QAPMCIOP	Communications IOP data (one per IOP)
QAPMDDI	Distributed Digital Interface (DDI) data (one per link)
QAPMDIOP	Storage device IOP data (one per IOP)
QAPMDISK	Disk storage data (one per read/write head)
QAPMDOMINO	Domino(TM) for iSeries(TM) data (one record per domino server)
» QAPMDPS	Data port services «
QAPMECL	Token-ring file entries (one per link)
QAPMETH	Ethernet statistics (one per link)
QAPMFRLY	Frame relay data (one per link)
QAPMHDLC	HDLC statistics (one per link)
QAPMHTTPB	Basic data for IBM(R) HTTP server (powered by Apache) (one per server)
QAPMHTTPD	Detailed data for IBM HTTP server (powered by Apache) (one per server component)
QAPMIDLC	Integrated services digital network data link control file entries (one per link)
QAPMIOPD	Extended IOP data » (network server and virtual I/O data) «
QAPMJOBMI	MI job data (one record per job, task, or thread). You might find information about task type extenders useful when using this document
QAPMJOBOS	Job operating system data (one record per job)
QAPMJOBS and QAPMJOBL	Job data (one record per job, task, or thread)
QAPMJOBWT	Job, task, and thread wait conditions
QAPMJOBWTD	A description of the counter sets found in file QAPMJOBWT.
QAPMJSUM	Job summary data by job group (one record per job group)
QAPMLAPD	Integrated services digital network LAPD file entries (one per link)
QAPMLIOP	Twinaxial workstation controller data (one per physical controller)
» QAPMLPAR	Logical partition (one record per logical partition) «
QAPMMIOP	Multifunction IOP (one per IOP)

File	Description
QAPMPOOL and QAPMPOOLL	Main storage data (one per system storage pool)
QAPMPOOLB	Storage pool data (one per pool)
QAPMPOOLT	Storage pool tuning data (one per pool)
QAPMPPP	Point-to-Point Protocol data (one per link)
QAPMRESP	Local workstation response time (one per workstation)
QAPMRWS	Remote workstation response time
QAPMSAP	TRLAN, Ethernet, DDI, and Frame Relay SAP file entries (one per SAP entry)
QAPMSNA	SNA data
QAPMSNADS	SNADS data (one per SNADS job)
QAPMSTND	DDI station data
QAPMSTNE	Ethernet station file entries
QAPMSTNL	Token-ring station file entries
QAPMSTNY	Frame relay station file entries
QAPMSYS and QAPMSYSL	System performance data
QAPMSYSCPU	System CPU usage data
QAPMSYSTEM	System-level performance data
QAPMTCP	TCP/IP data
QAPMTCPIFC	TCP/IP data for individual TCP/IP interfaces
QAPMUSRTNS	User-defined transaction data (Each job has one record for each type of transaction)
QAPMX25	X.25 statistics (one per link)

Performance data files: File abbreviations

The "Performance data files" on page 83 use abbreviations in the field and byte data tables. These abbreviations include:

Abbreviation	Description
Primary files	These files are related to and generated from the category.
C	Character in the Attributes column.
» **H**	Hexadecimal in the Attributes column. «
PD	Packed decimal in the Attributes column.
Z	Zoned decimal in the Attributes column.
IOP	Input/output processor or I/O processor. The processors that control the activity between the host system and other devices, such as disks, display stations, and communication lines.
DCE	Data circuit-terminating equipment.
MAC	Medium-access control. An entity in the communications IOP.
LLC	Logical link control. An entity in the communications IOP.
Beacon frame	A frame that is sent when the ring is inoperable.
Type II frame	A connection-oriented frame (information frame) used by Systems Network Architecture (SNA).
I-frame	An information frame.
» B	The DDS binary data type of 4 digits, which is 2 bytes, in the Attributes column. «

Performance data files: Collection Services system category and file relationships

When you collect performance data using “Collection Services” on page 32, the data is stored in a management collection (*MGTCOL) object. The CRTPFRDTA command exports data from that management collection object and then writes the data to the “Performance data files” on page 83. Each type of data that can be independently controlled and collected by Collection Services is represented by a data category. Each data category contains or provides data that is written to one or more performance data files. For a database file or member to be created, the category (or group of categories) that the file or member is dependent on must exist and be processed by CRTPFRDTA. The table below identifies the category-to-file relationships. There are three types of relationships:

Relationship	Description
Primary files	These files are related to and generated from the category.
Compatibility files	These files are logical files that join primary files to provide performance database compatibility with the previous file structure. If the system generates all participating files (primary categories), compatibility files are also generated.
Secondary files	These files are related to and contain some data that is derived from data contained in the category or in the primary file. However, they are not controlled by that category.

Users should note the following:

1. The CRTPFRDTA command generates a database file only when that file is a primary file for the selected category.
2. If a primary file is listed for multiple categories, you must select each of those categories to generate the file.
3. If a primary file for one category is listed as a secondary file for another category, you must select the second category to ensure complete information in your generated database file. For example, as shown in the table below, to generate a complete database file for QAPMECL, you must select both *CMNBASE and *CMNSTN.
4. The system generates compatibility files only when it generates all associated primary files.

The table below illustrates the relationships between system categories and performance database files.

Category	Primary files	Compatibility files	Secondary files
*SYSBUS	QAPMBUS		
*POOL	QAPMPOOLB	QAPMPOOLL	
*POOLTUNE	QAPMPOOLT	QAPMPOOLL	
*HDWCFG	QAPMHDWR		
*SUBSYSTEM	QAPMSBSD		
*SYSCPU	QAPMSYSCPU	QAPMSYSL	
*SYSLVL	QAPMSYSTEM	QAPMSYSL	
*JOBMI	QAPMJOBMI QAPMJOBWT QAPMJOBWTD QAPMJSUM	QAPMJOBL QAPMSYSL	QAPMSYSTEM
*JOBOS	QAPMJOBOS QAPMJSUM	QAPMJOBL QAPMSYSL	QAPMSYSTEM
*SNADS	QAPMSNADS		
*DISK	QAPMDISK		QAPMSYSTEM

*IOPBASE	QAPMIOPD QAPMLIOP QAPMDIOP QAPMCIOP QAPMMIOP		
*IPCS	QAPMIOPD QAPMTSK		
*CMNBASE	QAPMASYN QAPMBSC QAPMDDI QAPMECL QAPMETH QAPMFRLY QAPMHDLC QAPMIDLC QAPMLAPD QAPMPPP QAPMX25		
*CMNSTN	QAPMSTND QAPMSTNE QAPMSTNL QAPMSTNY none		QAPMDDI QAPMETH QAPMECL QAPMFRLY QAPMX25
*CMNSAP	QAPMSAP		
*LCLRSP	QAPMRESP		
*APPN	QAPMAPPN		
*SNA	QAPMSNA		
*EACACHE	none		QAPMDISK (see note)
*TCPBASE	QAPMTCP		
*TCPIFC	QAPMTCPIFC		
*DOMINO	QAPMDOMINO		
*HTTP	QAPMHTTPB QAPMHTTPD		
*USRTNS	QAPMUSRTNS		
*DPS	QAPMDPS		
*LPAR	QAPMLPAR		
Note: This category is not selectable by CRTPFRDTA. However, it causes additional data to be reported by the *DISK category.			

iSeries$^{(TM)}$ Navigator monitors

The monitors included in iSeries Navigator use “Collection Services” on page 32 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$^{(TM)}$ Navigator for Wireless" on page 132.

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:

> **"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.
>
> **"Configure a monitor" on page 89**
> 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.
>
> **"Scenarios: iSeries$^{(TM)}$ Navigator monitors" on page 89**
> This topic 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.

Monitor concepts

The system monitors display the data stored in the collection objects that are generated and maintained by "Collection Services" on page 32. 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” on page 95. 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 OS/400(R) 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.

Configure a monitor

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(TM) 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.

Scenarios: iSeries(TM) Navigator monitors

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(TM) Navigator monitors” on page 87.

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

“Scenario: System monitor” on page 90
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.

“Scenario: Job monitor for CPU utilization” on page 91
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.

»
“Scenario: Job monitor with Advanced Job Scheduler notification” on page 92
See an example job monitor that sends an e-mail to an operator when the threshold limit of a job is exceeded. «

“Scenario: Message monitor” on page 94
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.

Scenario: System monitor:

Situation

As a system administrator, you need to ensure that the iSeries(TM) 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 **OS/400(R) 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 **OS/400 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 **OS/400 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:

Situation

You are currently running a new application on your iSeries$^{(TM)}$ 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) For **Subsystem**, specify QINTER.
      3) 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 **OS/400(R) 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:

Situation

You are currently running an application on your iSeries(TM) 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:

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) For **Subsystem**, specify QINTER.
      3) 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 **OS/400$^{(R)}$ 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.

```
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.

Scenario: Message monitor:

Situation

You company has several iSeries(TM) 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 OS/400(R) commands when specified messages are received.

Graph history

Graph history provides a graphical view of performance data collected over days, weeks, months, or years with "Collection Services" on page 32. 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.

- **"Graph history concepts"**
  The amount of historical data available to graph history depends largely on the collection retention value in Collection Services, and on whether "IBM Performance Management for eServer iSeries" on page 96(TM) is enabled. See this topic for a description of the available options for managing and displaying records of performance data.
- **"Use graph history"**
  Graph history is accessible through iSeries Navigator. See this topic for step-by-step instructions.

For more information about monitoring system performance, see the "Track performance" on page 15 topic.

Graph history concepts

Graph history displays data contained in the collection objects created by "Collection Services" on page 32. 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(TM) Navigator to "Activate PM iSeries" on page 99 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, 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**
  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**
  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**
  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.

Use graph history

Graph history is included in iSeries(TM) Navigator. To view the graph history of the data that you are monitoring with "Collection Services" on page 32, 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(R) Performance Management for eServer(TM) 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.

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.

IBM Performance Management for eServer iSeries

IBM$^{(R)}$ Performance Management for eServer$^{(TM)}$ iSeries$^{(TM)}$ (PM @server iSeries or 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 “PM iSeries reports” on page 111 and graphs that show your server’s growth and performance. You can access your reports electronically using a traditional browser. These reports can help you plan for and manage system resources through ongoing analysis of key performance indicators.

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 also helps you better understand (by using capacity planning) how your business trends relate to the timing of required hardware upgrades such as CPU or disk. Capacity planning information is provided by trending system utilization resources and throughput data, which can be thought of as an early warning system for your servers. Think of PM iSeries as a virtual resource that informs you about the “health” of your system.

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

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

“Configure PM iSeries” on page 98
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.

“Manage PM iSeries” on page 108
Now that you have set up your network, you can perform a variety of tasks with PM iSeries.

“PM iSeries reports” on page 111
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.

PM iSeries concepts

PM iSeries$^{(TM)}$ uses Collection Services to gather the nonproprietary performance and capacity data from your server and then sends the data to IBM$^{(R)}$. 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.

“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.

“Data collection considerations for PM iSeries”
PM iSeries uses “Collection Services” on page 32 to gather performance data. Learn how PM iSeries and Collection Services work together to provide the data you need.

Benefits of PM iSeries: When you use the PM iSeries(TM) 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.

See “IBM Performance Management for eServer iSeries” on page 96 to learn more about what you need to do before using PM iSeries.

Operational Support Services for PM iSeries offering: 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(TM) 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.

See “IBM Performance Management for eServer iSeries” on page 96 to learn more about what you need to do before using this function.

Data collection considerations for PM iSeries: 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(TM) 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.

If you do not want to start Collection Services, you can “Turn off PM iSeries momentarily” on page 110.

Configure PM iSeries

PM iSeries(TM) 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” on page 99
PM iSeries ships with OS/400(R), but you must activate it to use its collecting capabilities.

“Determine which PM iSeries transmission method to use” on page 99
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.

“Customize PM iSeries” on page 106
Now that you have set up your network, you may need to customize PM iSeries to fit your needs.

Activate PM iSeries: You must start PM iSeries$^{(TM)}$ 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" on page 95 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$^{(TM)}$ 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 PM iSeries transmission method to use"$^{(R)}$.

For an overview of iSeries performance topics, see "Performance," on page 1.

Determine which PM iSeries transmission method to use: Since V5R1, the PM iSeries$^{(TM)}$ 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 PM iSeries$^{(TM)}$ data with Service Agent over Extreme Support (Universal Connection)" on page 100
  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.
- "Sending data with SNA protocol" on page 100
  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 OS/400$^{(R)}$ 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" on page 108.

Send PM iSeries(TM) 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(TM) over Extreme Support to send the data to IBM(R).

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" on page 99
   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 "Sending data with SNA protocol."

Once you have configured PM iSeries, you are ready to do the other tasks to "Manage PM iSeries" on page 108.

Gather PM iSeries performance data: You can use Management Central to gather your PM iSeries(TM) performance data if you have done the following tasks:

1. "Activate PM iSeries" on page 99
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" on page 108.

Sending data with SNA protocol: If you choose not to take advantage of "Send PM iSeries(TM) data with Service Agent over Extreme Support (Universal Connection)," you can still use the character-based interface to transmit data. PM iSeries(TM) 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" on page 99
   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(R) by using "Set a direct dial line for PM iSeries" on page 107, 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 "PM iSeries network for a single server" that sends its data directly to IBM.
   - As a "PM iSeries network for a host server" on page 102, 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 "PM iSeries network for a remote server" on page 102, 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" on page 103
   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" on page 106
   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 "Manage PM iSeries" on page 108.

PM iSeries network for a single server: A single server sends its data directly to IBM(R). Here are the steps that you need to follow to configure PM iSeries(TM) 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 "Sending data with SNA protocol" on page 100.

Once you have configured your servers, you are ready to do the other tasks to "Manage PM iSeries" on page 108.

PM iSeries network for a host server: A host server receives performance data from other servers and then forwards the data to IBM(R). Here are the steps that you need to follow to configure PM iSeries(TM) for a host server only if PM iSeries gathers data and transmits data over SNA:

1. From the Configure PM eServer(TM) 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 "Sending data with SNA protocol" on page 100.

Once you have configured your servers, you are ready to do the other tasks to "Manage PM iSeries" on page 108.

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(TM) 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 "Sending data with SNA protocol" on page 100.

Once you have configured your servers, you are ready to do the other tasks to "Manage PM iSeries" on page 108.

Work with remote servers: In some sites, a host server in a network sends the required performance data to IBM^(R) 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^(TM) 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^(TM) 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 "Schedule jobs with PM iSeries" on page 109

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 PM iSeries" on page 105 for information on how to define your controller descriptions. If your network does not meet these assumptions, see "Work with remote servers in a non-APPN network" 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" on page 106 to use a specific line connection.

Work with remote servers in a non-APPN network: The primary server receives PM iSeries^(TM) data from other servers and then sends the data to IBM^(R). 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:

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:

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.

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$^{(TM)}$ iSeries Menu to add Q1PRMxxx as a remote server. See “Work with remote servers” on page 103 for instructions on how to add a remote server.

Now that you have finished configuring PM iSeries, see “Manage PM iSeries” on page 108 for other tasks that you can perform with PM iSeries.

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$^{(TM)}$ 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. Refer to the following commands to determine how these parameters are defined on the system:

- Create Controller Description (APPC) (CRTCTLAPPC) command
- Change Controller Description (APPC) (CHGCTLAPPC) command
- Display Controller Description (DSPCTLD) command

For systems that are not configured to use APPN, see “Work with remote servers in a non-APPN network” on page 103 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)
```

Now that you have finished configuring PM iSeries, see “Manage PM iSeries” on page 108 for other tasks that you can perform with PM iSeries.

Customize PM iSeries: The Work with PM eServer(TM) iSeries(TM) 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(R) 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” on page 107
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” on page 97.

See “Manage PM iSeries” on page 108 for other tasks that you can perform with PM iSeries.

Verify the PM iSeries data number: If your server is using a direct dial connection to IBM(R), you must verify that the PM iSeries(TM) 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" on page 108 for the tasks that you can perform.

Set a direct dial line for PM iSeries: For most locations, PM iSeries(TM) tries to select the correct data telephone number for your location. You should always "Verify the PM iSeries data number" on page 106 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(R) 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(TM) 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.

Vary a line off and on with PM iSeries: Sometimes the line that PM iSeries(TM) uses is in the connect pending state. This state does not allow PM iSeries to access the line to transmit data. The PM eServer(TM) 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/400 line control by using the Configure PM eServer iSeries (CFGPM400) command.

See "Manage PM iSeries" for additional tasks that you can perform with PM iSeries.

Manage PM iSeries

After you have set up your network to use PM iSeries(TM), you can perform the following tasks:

"Deactivate PM iSeries"
Learn how you can stop PM iSeries.

"Change PM iSeries contact information" on page 109
Learn how to change your contact information from the original settings.

"Schedule jobs with PM iSeries" on page 109
Learn how to schedule jobs with PM iSeries.

"Omit items from PM iSeries analysis" on page 110
Learn how to omit jobs, users, and communications lines when performing an analysis with PM iSeries.

"Turn off PM iSeries momentarily" on page 110
Learn how you can stop PM iSeries momentarily.

"Display PM iSeries status" on page 110
Learn how to use iSeries Navigator or the PM eServer(TM) iSeries menu to display PM iSeries status.

"View PM iSeries reports" on page 111
See examples of the PM iSeries reports and explanations of how to interpret those reports.

"Graph history" on page 95
Graph history provides a graphical view of performance data that was collected over a specified period of time. Find out how to view this data.

Deactivate PM iSeries: To stop PM iSeries(TM) 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.

See "Manage PM iSeries" on page 108 to learn about other tasks that you can perform.

Change PM iSeries contact information: During the configuration of the PM iSeries(TM) 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(TM) iSeries Menu and press Enter. The Work with Contact Information display appears.
3. Change the contact information, as appropriate, and press Enter.

See "Manage PM iSeries" on page 108 for other tasks that you can perform.

Schedule jobs with PM iSeries: Integral to the PM iSeries(TM) 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(TM) 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(R). 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.
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.

See "Manage PM iSeries" on page 108 to learn about other tasks that you can perform.

Omit items from PM iSeries analysis: The PM iSeries(TM) 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(TM) 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.

See "Manage PM iSeries" on page 108 to learn about other tasks that you can perform.

Turn off PM iSeries momentarily: If you need to stop PM iSeries(TM) 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.

See "Schedule jobs with PM iSeries" on page 109 to learn about other things that you can do with the scheduler.

See "Manage PM iSeries" on page 108 to learn about other tasks that you can perform.

Display PM iSeries status: You can use either iSeries(TM) Navigator or the PM eServer(TM) iSeries Menu on your server to display the status of PM iSeries. Use the IBM(R) 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.

See "Manage PM iSeries" on page 108 to learn about other tasks that you can perform.

View PM iSeries reports: The output of the PM iSeries$^{(TM)}$ offering is a set of management reports and graphs on a monthly or quarterly basis. The PM iSeries offering has two "Operational Support Services for PM iSeries offering" on page 97 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. To view each report and graph in detail and learn about some of their benefits and uses, see the PM eServer iSeries Web site.

See "Manage PM iSeries" on page 108 to learn about other tasks that you can perform.

PM iSeries reports

The server automatically records various statistics about its operating environment during the course of normal operation. "Collection Services" on page 32 has the capability to consolidate these statistics. "IBM Performance Management for eServer iSeries" on page 96 can collect and transmit these statistics to IBM$^{(R)}$, which forms the basis for all PM iSeries$^{(TM)}$ 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.

Performance Tools

The Performance Tools for iSeries$^{(TM)}$ 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" on page 32 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" on page 113 topic.

“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.

“Install and configure Performance Tools” on page 118
See this topic for installation and setup instructions.

“Performance Tools reports” on page 118
Performance Tools reports provide information on data that has been collected over time. Use these reports to get additional information about the performance and use of system resources.

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.

Performance Tools concepts

The Performance Tools for iSeries(TM) licensed program analyzes two distinct types of performance data: sample data and trace data. “Collection Services” on page 32 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.

“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. See this topic for a description of the functions included in this licensed program.

“Manager and Agent feature comparison” on page 113
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 plug-in” on page 113
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.

»

“Reporting CPU utilization” on page 114
Find out how the total CPU that is consumed across virtual processors is reported.

“Reporting configured capacity” on page 116
Find out where the information for configured capacity is recorded.

“5250 online transaction processing (OLTP)” on page 117
Find out what is meant by 5250 online transaction processing and what jobs or threads are associated with this work. «

Functions provided in Performance Tools: 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.
"Performance Tools plug-in"	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$^{(TM)}$ Navigator.
"Performance Tools reports" on page 118	The reports organize Collection Services performance 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" on page 121	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 .

Manager and Agent feature comparison: 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" on page 121 reporting function and its associated commands are included in the base option in the Performance Tools for iSeries$^{(TM)}$ 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 and access group), system activity monitoring, and performance graphics.

Performance Tools plug-in: Performance Tools can display performance data from the Display Performance Data graphical user interface (GUI), which is a plug-in for iSeries$^{(TM)}$ 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. For more information on the Manager feature, see the “Manager and Agent feature comparison” on page 113 topic.

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.

Reporting CPU utilization: » 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(R) 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.

Reporting configured capacity: The partition capacity values are determined initially when the partition is started through the Hardware Management Console (HMC) using a Configuration Profile for the partition and depends on the capacity resources available at the time. These initial values can be altered through configuration changes while the partition is active. For more information about logical partitions, see the shared processor pool topic.

Logical partitions 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 iSeries(TM) performance database files in a 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 iSeries performance database files 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 iSeries performance database files in a column named SYSPLU.

Partition guaranteed capacity
Processor capacity configured to a shared processor logical partition from the shared processor pool. This value is included in the iSeries performance database files in a column named SYSCTA. The 5250 OLTP capacity configured is recorded in column named SYIFTA.

Partition processor utilization
Processor utilization by a logical partition. In a shared processor logical partition with uncapped capacity, this value may exceed the guaranteed capacity if there is unused capacity in the shared processor pool. This value is included in the iSeries performance database files in a column named SYSPTU. The 5250 OLTP capacity used is recorded in column named SYIFUS.
The maximum processor capacity in a partition is determined by the number of virtual processors configured.

Partition available capacity
The amount of processor capacity that could have been used by the logical partition. This value is included in the iSeries performance database files 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.

5250 online transaction processing (OLTP): The definitions for 5250 interactive and batch type work changed in V5R3. The term 5250 online transaction processing (OLTP) replaces the term 5250 interactive when discussing CPU utilization and system resource consumption. The new term recognizes the fact that interactive is a more generic description of an application interface that is characterized by user input and the associated responses. As a result, interactive can refer to:

- The familiar iSeries(TM) interactions through a 5250 session, a pass-through job, or a Telnet job.
- A workstation-based request from a Domino(R) mail or calendar or a browser-based application.
- Workloads that were historically considered interactive, such as, DDM.

Prior to V5R3, all OS/400(R) jobs are categorized into one of ten categories based on certain job attributes, for example, DDM, CA4, and INT. Based on these categories, Performance Tools reports these jobs as interactive or batch.

This approach for showing interactive and batch CPU utilization was not meaningful in the past because some of the processor utilization in some categories was shown as belonging to both interactive and batch. For example, iSeries Access jobs use both interactive and batch, depending on the function. Prior to V5R3, these jobs were included in the CA4 category and listed as interactive. The DDM server jobs were also listed as interactive.

Beginning in V5R3, an approach is implemented in the Performance Tools licensed program to better distribute the workloads, depending on which processor capacity feature the CPU cycles were charged against. The interactive CPU reporting now means those jobs whose CPU is allocated to the 5250 OLTP processor capacity. As a result, the iSeries Access jobs are listed in the appropriate sections of the Performance Tools reports. In addition, the DDM jobs move from the Interactive workload section of the reports to the Non-interactive workload section.

«

Install and configure Performance Tools

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 📖.

Performance Tools reports

The Performance Tools reports provide an easy way for you to look at your collected data and isolate performance problems. After you have collected performance data over time, you can "Print the performance reports" on page 120 the reports to see how and where system resources are being used. The reports can direct you to specific application programs, users, or inefficient workloads that are causing slower overall response times.

"Collection Services" on page 32 provides data for most of the Performance Tools reports with the exception of the Transaction, Lock, and Trace reports. You must use the STRPFRTRC and ENDPFRTRC commands to collect the "Collect information about an application's performance" on page 19 for those three reports.

The following list describes each report, gives a brief overview as to why you would use a particular report, and links to samples of each report. In addition, each report is discussed in detail in the Performance Tools book 📖.

Overview of Performance Tools reports

Report	Description	What is shown	How you use the information
System	Uses Collection Services data to provide an overview of how the system is operating. The report contains summary information on the workload, resource use, storage pool utilization, disk utilization, and communications. Run and print this report often to give you a general idea of your system use.	System workload. The report includes the database capabilities data.	Workload projection
Component	Uses Collection Services data to provide information about the same components of system performance as a System Report, but at a greater level of detail. This report helps you find which jobs are consuming high amounts of system resources, such as CPU, disk, and so on.	Resource use, communications, system and user jobs. The report includes the database capabilities data and the Interactive Feature utilization.	Hardware growth and configuration processing trends
Transaction	Uses trace data to provide detailed information about the transactions that occurred during the performance data collection.	Workload and utilization of CPU, disk, main storage, transaction workload, object contention	Workload projection, pool configuration, application design, file contention, and program use
Lock	Uses trace data to provide information about lock and seize conflicts during system operation. With this information you can determine if jobs are being delayed during processing because of unsatisfied lock requests or internal machine seize conflicts. These conditions are also called waits. If they are occurring, you can determine which objects the jobs are waiting for and the length of the wait.	File, record, or object contention by time; the holding job or object name; the requesting job or object name	Problem analysis. Reduction or elimination of object contention.
Batch Trace Job	Uses trace data to show the progression of different job types (for example, batch jobs) traced through time. Resources utilized, exceptions, and state transitions are reported.	Job class time-slice end and trace data	Problem analysis and batch job progress

Job Interval	Uses Collection Services data to show information on all or selected intervals and jobs, including detail and summary information for interactive jobs and for noninteractive jobs. Because the report can be long, you may want to limit the output by selecting the intervals and jobs you want to include.	Jobs by interval	Job data
Pool Interval	Uses Collection Services data to provide a section on subsystem activity and a section on pool activity. Data is shown for each sample interval. Because the report can be long, you may want to limit the output by selecting the intervals and jobs you want to include.	Pools by interval	Pool data
Resource Interval	Uses Collection Services data to provide resource information on all or selected intervals. Because the report can be long, you may want to limit the output by selecting the intervals you want to include.	Resources by interval	System resource use

Performance explorer and Collection Services are separate collecting agents. Each one produces its own set of database files that contain grouped sets of collected data. You can run both collections at the same time.

For a list of reports for other tools, see the following:

- “Performance explorer reports” on page 127
- “PM iSeries reports” on page 111

Print the performance reports: You can print reports using the performance data that you collected. Prior to V5R1, Option 3 (Print performance report) displayed a list of performance members that were located in the QAPMCONF file. This list included both sample data and trace data that was collected by the Start Performance Monitor (STRPFRMON) command. Collection Services does not collect trace data. However, you can use the STRPFRTRC and TRCINT commands to collect trace data. This data is located in the QAPMDMPT file. Therefore, in V5R1 and later, you see two views of the Print Performance Report display, one for sample data and one for trace data.

Note: If your trace data and sample data are both in the current library, you can use F20 to toggle between the two Print Performance Report displays.

After you have collected your data, you must create a set of performance data files from the performance information stored in a management collection (*MGTCOL) object. Use the Create Performance Data (CRTPFRDTA) command. After you have created the data files, you can request to print your reports.

Use the following commands to print reports for sample data that you collected with Collection Services:

- Print System Report (PRTSYSRPT)
- Print Component Report (PRTCPTRPT)
- Print Job Interval Report (PRTJOBRPT)
- Print Pool Report (PRTPOLRPT)
- Print Resource Report (PRTRSCRPT)

Use the following commands to print reports for trace data that you collected with the Start Performance Trace (STRPFRTRC) and Trace Internal (TRCINT) commands:

- Print Transaction Report (PRTTNSRPT)
- Print Lock Report (PRTLCKRPT)
- Print Job Trace Report (PRTTRCRPT)

Note: You must use the End Performance Trace (ENDPFRTRC) command to stop the collection of performance trace data and then optionally write performance trace data to a database file before you can print the Transaction reports.

Performance explorer

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 OS/400(R) licensed program. The reporting function and its associated commands are part of the base option in the Performance Tools for iSeries(TM) licensed program and therefore, are available with either the Manager feature or the Agent feature. The AS/400(R) 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.

To learn more about performance explorer, refer to any of the following performance explorer topics.

"Performance explorer concepts" on page 122
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.

“Configure performance explorer” on page 127
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 reports” on page 127
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.

For more detailed information, refer to the Performance Tools book .

Performance explorer concepts

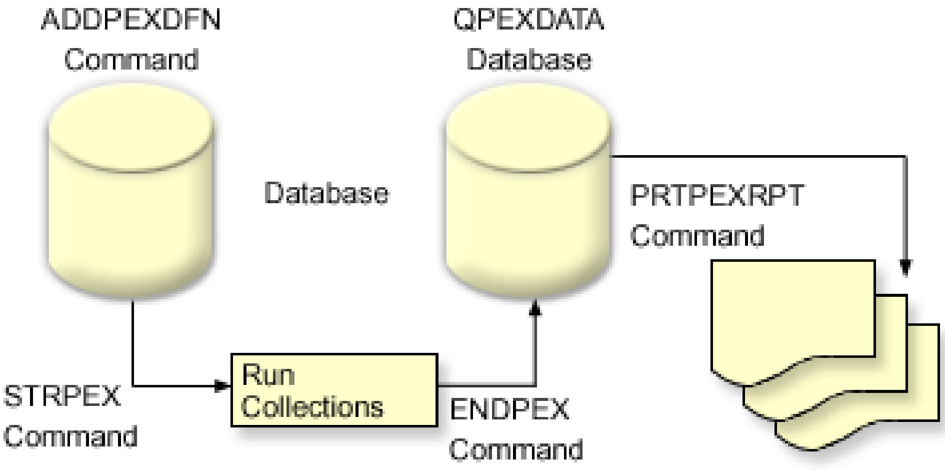
Like “Collection Services” on page 32, performance explorer collects data for later analysis. However, they collect very different types of data. Collection Services collects a broad range of system data at regularly schedules intervals, with minimal system resource consumption. In contrast, performance explorer starts a session that collects trace-level data. This trace generates a large amount of detailed information about 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$^{(TM)}$ 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.

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” on page 127. 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.

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

"Performance explorer database files" on page 125
The data that performance explorer collects is stored in performance explorer database files.

"Performance explorer benefits" on page 127
Performance explorer contains a variety of functions that can help gather and analyze detailed performance information. This topic provides an overview of those various functions.

Performance explorer definitions: 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(R) 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 OS/400(R) 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 that consume excessive CPU utilization based on source program statement numbers. You can also identify a program that is constantly branching between the start of the program and subroutines at the end of the program. If the program is large enough, this constant jumping back and forth can cause excessive page fault rates on a system with limited main storage.

- 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, OS/400 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(TM) 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. */
```

Performance explorer database files: The following table shows the performance explorer 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
Reference information	QAYPEREF
General information	QAYPERUNI
PMC selection	QAYPEFQCFG
Basic configuration information	QAYPECFGI
Machine interface (MI) complex instructions collected on	QAYPELCPLX
Jobs collected on	QAYPELJOB
Metrics to collect data on	QAYPELMET
Machine interface (MI) program, module, or procedures collected on	QAYPELMI

Type of information contained in file	File name
Licensed Internal Code (LIC) modules to collect data on	QAYPELLIC
Task names to collect data on	QAYPELNAMT
Task number to collect data on	QAYPELNUMT
Machine interface (MI) complex instructions mapping	QAYPEMICPX
Event type and subtype mapping	QAYPEEVENT
Hardware mapping data	QAYPEHWMAP
Licensed Internal Code (LIC) address resolution mapping	QAYPEPROCI
Segment address resolution mapping	QAYPESEGI
Process and task resolution mapping	QAYPETASKI
Common trace data for all events	QAYPETIDX
Auxiliary storage management event data	QAYPEASM
Base event data	QAYPEBASE
Disk event data	QAYPEDASD
Disk server event data	QAYPEDSRV
Page fault event data	QAYPEPGFLT
Resource management process event data	QAYPERMPM
Resource management seize lock event data	QAYPERMSL
Advanced 36(R) event data	QAYPES36
Segment address range (SAR) data	QAYPESAR
Unknown event data	QAYPEUNKWN
Basic statistics data	QAYPESTATS
Statistic profiling summary data	QAYPEPSUM
Licensed Internal Code (LIC) bracketing data	QAYPELBRKT
Machine interface (MI) user event data	QAYPEMIUSR
Machine interface (MI) program bracketing data	QAYPEMBRKT
Addresses of machine interface (MI) pointer	QAYPEMIPTR
User-defined bracketing hook data	QAYPEUSRDF
Hardware monitor data	QAYPEHMON
Hardware monitor total data	QAYPEHTOT
Release, version, modification level	QRLVRM
Performance explorer level indicator	QRLLVL
Performance explorer Java(TM) event data	QAYPEJVA
Performance explorer Java class information data	QAYPEJVCI
Performance explorer Java method information data	QAYPEJVMI
Performance explorer Java name information data	QAYPEJVNI
Synchronization event data	QAYPESYNC
Communications event data	QAYPECMN
File Serving event data	QAYPEFILSV
Heap event data	QAYPEHEAP
PASE event data	QAYEPASE
Trace job equivalent event data	QAYPETBRKT

Type of information contained in file	File name
Task switch event data	QAYPETSKSW
Synchronization event data	QAYPESYNC
Program profile data	QAYPEPPANE

Performance explorer reports: Performance explorer gathers detailed information about a program or job's behavior and performance and stores this information in "Performance explorer database files" on page 125. 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" on page 123 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 book .

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)
```

Performance explorer benefits: Performance explorer has advantages for people who need detailed performance analysis on an iSeries$^{(TM)}$ 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.

Configure performance explorer

To configure performance explorer, follow these steps:

1. "Performance explorer definitions" on page 123 that informs the iSeries$^{(TM)}$ 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 (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 (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).

4. Run your command, program, or workload for data that you want to analyze.
5. "Ending performance explorer" 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 other option for analysis is to write your own queries over the set of database files.

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 OS/400(R) operating system, except the Print Performance Explorer Report (PRTPEXRPT) command.
- The Performance Tools menu options.

To see a performance explorer work cycle, see "Performance explorer concepts" on page 122.

Ending performance explorer

To end the performance explorer session, use the End Performance Explorer (ENDPEX) command. The 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(R)-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

iDoctor for iSeries(TM) is a suite of tools consisting of these components: Consulting Services, Job Watcher, Java(TM) 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$^{(TM)}$ Watcher
Java Watcher provides invaluable information to aid in debugging some of the most complex problems in the area of Java and WebSphere$^{(R)}$.

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.

Performance Trace Data Visualizer (PTDV)

The Performance Trace Data Visualizer (PTDV) is a Java$^{(TM)}$ application that can be used for performance analysis of applications running on iSeries$^{(TM)}$ servers. PTDV works with performance explorer in the OS/400$^{(R)}$ 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$^{(R)}$ 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.

Performance Management APIs

The performance management APIs allow you to collect and manage performance data using Collection Services, performance collector, performance explorer, and PM iSeries$^{(TM)}$.

The Performance Management APIs include:

- Collection Services APIs
- Performance Collector APIs
- Performance Explorer (PEX) APIs
- IBM$^{(R)}$ Performance Management for eServer$^{(TM)}$ iSeries (PM iSeries) APIs

“Work with” commands for OS/400 performance

OS/400$^{(R)}$ includes a number of commands that can 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$^{(TM)}$ Navigator, see “iSeries$^{(TM)}$ Navigator monitors” on page 87.

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.

Command	Function
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.

Extended Adaptive Cache

Improve your iSeries(TM) system performance with Extended Adaptive Cache! Extended Adaptive Cache is an advanced large 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 generates statistical information for the data and then uses a mix of management strategies to determine which data to cache. Extended Adaptive Cache has proven to be highly effective on many types of workloads.

To learn more, keep reading:

- "Extended Adaptive Cache Concepts"
  Explore Extended Adaptive Cache. Find information about planning, restrictions, and important considerations before you begin to use this tool.
- **"Restrictions and considerations for Extended Adaptive Cache" on page 131**
  See what components Extended Adaptive Cache requires and learn more about what to expect.

Extended Adaptive Cache Concepts

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(R) 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(TM) 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.

To learn more about what components are required, see **“Restrictions and considerations for Extended Adaptive Cache”**

Restrictions and considerations for Extended Adaptive Cache: Before you begin using “Extended Adaptive Cache Concepts” on page 130, 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$^{(TM)}$ 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.

Start Extended Adaptive Cache: To start “Extended Adaptive Cache Concepts” on page 130 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.

To find out whether your iSeries$^{(TM)}$ system is capable of using Extended Adaptive Cache, see “Restrictions and considerations for Extended Adaptive Cache.”

Get Extended Adaptive Cache

After 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(TM) hardware is sold, or contact your local IBM(R) representative.

Workload Estimator for iSeries

The Workload Estimator helps you size system needs based on estimated workloads for specific workload types. PM iSeries(TM) is an integrated OS/400(R) function that users under processor warranty or on an IBM(R) 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(TM), Java(TM), and WebSphere(R), or the consolidation of multiple AS/400(R) or iSeries traditional OS/400 workloads on one system. This capability allows you to plan for future system requirements based on existing utilization data coming from your own system.

iSeries(TM) Navigator for Wireless

iSeries Navigator for Wireless lets you remotely monitor system performance and status using an Internet-ready telephone, a personal digital assistant (PDA) with a wireless modem, or a traditional Web browser. 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(R) servers

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.

PATROL for iSeries (AS/400) - Predict

The PATROL for iSeries(TM) (AS/400(R)) - 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.

Scenarios: Performance

One of the best ways to learn about performance management is to see examples illustrating how many of the applications and functions can be used in a sample business environment. Use the following scenarios and configuration examples to learn more about managing performance.

“Scenario: Improve system performance after an upgrade or migration” on page 21
In this scenario, you have just upgraded or migrated your system and it now appears to be running slower than before. This scenario helps you identify and fix your performance problem.

“Scenario: System monitor” on page 90
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.

“Scenario: Message monitor” on page 94
See an example message monitor that displays any inquiry messages for your message queue that occur on any of your iSeries(TM) servers. The monitor opens and displays the message as soon as it is detected.

“Scenario: Job monitor for CPU utilization” on page 91
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.

» **“Scenario: Job monitor with Advanced Job Scheduler notification” on page 92**
See an example job monitor that sends an e-mail to an operator when the threshold limit of a job is exceeded. «

Related information

Listed below are the iSeries(TM) manuals (sometimes called “white books”) and IBM(R) Redbooks(TM), in PDF format, that relate to the Performance topic. You can also view or print any of the following 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.
- **Web sites**
  - **iSeries Performance Capabilities Reference**

This reference provides highly technical information about server performance useful for performance benchmarking, capacity planning, and planning for server performance.

  - » **Three-In-One Benchmark**

    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**

    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. «

- **Redbooks:**
  - **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.

  - **Lotus(R) Domino(R) for AS/400(R): 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.

  - **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.

  - **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.

  - **Java(TM) and WebSphere(R) 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.

  - » **Managing OS/400(R) 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.«

- **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.

- **DB2(R) 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.

- » **Performance Management Tools**

  This IBM Redpaper is designed to help you understand the different iSeries performance management tools at the IBM i5/OS V5R3M0 level, that are available to you and when to use them. «

For complete information about iSeries performance, be sure to see the “Performance,” on page 1 topic.

Appendix. Notices

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

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

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

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

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

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

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

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

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. 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 INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

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

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

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

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
AIX
AIX 5L
Domino
e(logo)server
eServer
Operating System/400
OS/400
IBM
iSeries
pSeries
xSeries

Lotus, Freelance, and WordPro are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both.

Java$^{(TM)}$ and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux$^{(TM)}$ is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Terms and conditions for downloading and printing publications

Permissions for the use of the information you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce this information for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of this information, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display this information solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of this information, or reproduce, distribute or display this information 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 information or any 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 information 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 THIS INFORMATION. THE INFORMATION IS 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.

All material copyrighted by IBM Corporation.

By downloading or printing information from this site, you have indicated your agreement with these terms and conditions.

Code disclaimer information

IBM$^{(R)}$ 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. 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 INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Printed in USA