ivanti

Service and Asset Manager

Performance Tuning Guide

2019.3.1

Contents

About This Guide

- "Intended Audience" below

- "List of Topics" below

- "Related Documentation" on the next page

- "How to Contact Us" on the next page

The *Performance Tuning Guide for Service and Asset Manager* describes how to configure Service and Asset Manager including the platform, the Service and Asset Manager application server, and the Service and Asset Manager application database, to optimize performance for Service and Asset Manager.

Intended Audience

This document is intended for system administrators.

To perform the procedures described in this document you should have:

- Expert knowledge of Service and Asset Manager, platform, business objects, definition sets, and the Ivanti Service Manager Configuration Console. The Configuration Console is the interface where you apply changes to the databases to give structure to Ivanti Service Manager.

- Working knowledge of database structure and terminology.

- Working knowledge of the Microsoft Windows operating system.

- Working knowledge of the Microsoft .NET architecture.

- Working knowledge of Microsoft SQL Server Management Studio.

List of Topics

This document contains the following sections:

- "Server and Connection Tuning" on page 6: Describes how to configure servers and the connections between them in Ivanti Service Manager to optimize performance.

- "Platform and Application Tuning" on page 9: Describes how to configure Ivanti Service Manager and its platform to optimize performance.

- "Database Tuning" on page 19: Describes how to tune the Ivanti Service Manager databases for optimal performance.

- "Performance Monitoring" on page 35: Lists common performance monitoring parameters.

- "Troubleshooting" on page 42: Describes how to diagnose and fix common performance issues.

- "Deleting Inactive Tables and Records" on page 43: Describes how to delete inactive tables currently residing in the Ivanti Service Manager databases.

Related Documentation

Ivanti Service Manager has online help available within the application.

Additional documentation is available through

- The Ivanti community website. You may need to request user access if you cannot log in.

  Or through

- The Ivanti Product Documentation website. Click the Service Manager tile to see a list of the documents available.

How to Contact Us

To contact us about the documentation, or if you have any other questions or issues about Ivanti Service Manager, contact Ivanti Software Global Support services by logging an incident via Self Service at: https://www.ivanti.com/support/ivanti-support.

Server and Connection Tuning

This section describes how to configure servers and the connections between them in Ivanti Service Manager to optimize performance.

- "Ensuring that Hardware and Software Prerequisites are Met" below

- "Optimizing Network Utilization and Bandwidth" below

- "Deactivating Unused Web Servers in the Ivanti Service Manager Configuration Database" below

- "Working with Log Files and Logging Levels" on the next page

- "Configuring the Application Pool Recycling Schedule" on the next page

Ensuring that Hardware and Software Prerequisites are Met

Hardware and software requirements for Ivanti Service Manager are listed in the *System Requirements and Compatibility Matrix for Ivanti Service Manager*. Verify that all components in your environment meet those requirements.

Optimizing Network Utilization and Bandwidth

Recommendations for optimal Ivanti Service Manager performance are:

- Latency of 110 ms or less. See "Checking Database Latency and Moving Indexes to a Separate Disk" on page 21 for an example Microsoft SQL script that you can execute to check latency.

- A minimum of 1.5m bits/second in bandwidth between the Ivanti Service Manager application server and remotely located client computers.

Deactivating Unused Web Servers in the Ivanti Service Manager Configuration Database

To improve performance, deactivate any unused web servers from the Ivanti Service Manager configuration database. Follow these steps:

1. Log into the Ivanti Service Manager configuration database.
2. Open the **Web Servers** workspace.
3. Click **List View** to see a list of all of the web servers.
4. Open the web server to deactivate.
5. Uncheck **Server is Active**.
6. Click **Save**.

Working with Log Files and Logging Levels

Log files for all services, such as workflow and escalation, can provide valuable information for performance tuning.

We recommend that you set the log levels to **error**. You can set the log level to **debug** for troubleshooting, but it is very verbose so we do not recommend using that level on a regular basis.

Follow these steps to change the log level for the **WorkflowService** and **EscalationService** logs:

1. Log in to Ivanti Service Manager.

2. Open the **Logging Configuration** workspace.

3. Open an entry.

4. From the **Log Level** field, select a log level.

5. Click **Save**.

Configuring the Application Pool Recycling Schedule

We recommend that you schedule the application pool recycling to occur at a specific time rather than at an interval. Setting a specific time lets you schedule the recycling for off-peak times when performance is least likely to be impacted.

To configure the application pool recycling schedule, do the following:

1. Open the **Computer Management** panel by going to **Start > Control Panel > Administrative Tools > Computer Management**.

2. In the **Computer Management** panel, double-click **Services and Applications** and double-click **Internet Information Services (IIS)**.

3. In the **Connections** panel, double-click **Application Server** and double-click **Application Pools**.

4. In the list of application pools, highlight an application pool and click **Recycling...** located in the **Edit Application Pool** area.

Configuring the Application Pool Recycling Schedule
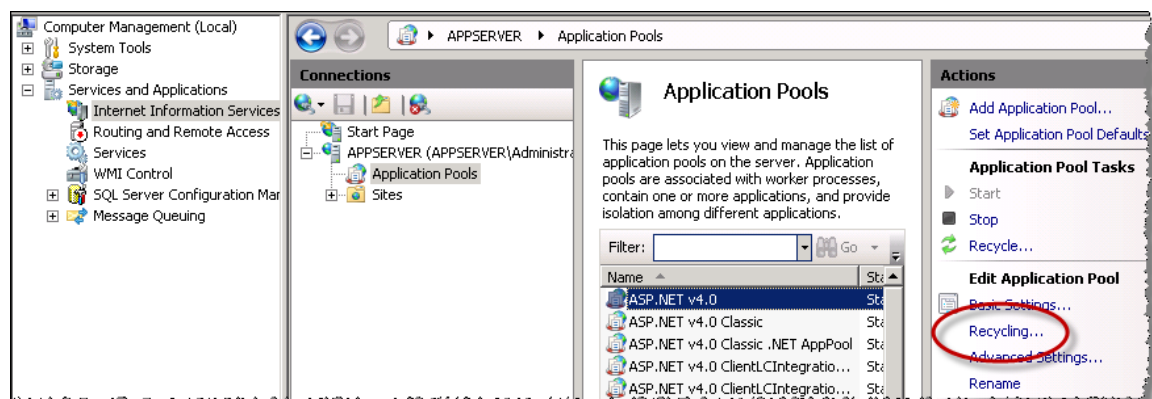
5. In the **Recycling Conditions** dialog box, check **Specific time(s)** and specify an off-hours time for the recycling to occur.

6. Repeat step 4 and step 5 to specify recycling schedules for the other application pools.

Platform and Application Tuning

This section describes how to configure Ivanti Service and Asset Manager and its platform to optimize performance.

- "About the WAN Environment" below

- "Planning and Scheduling Maintenance" on the next page

- "About Reviewing, Closing, and Deleting Records" on page 12

- "Tuning Business Rules and Workflows" on page 12

- "Best Practices for Dashboards and Layouts" on page 13

- "Improving Back-End Processing" on page 14

- "Using Event and Error Log Files for Troubleshooting" on page 15

- "About Using Combo Boxes Sparingly" on page 15

- "Evaluating Slow Running Queries " on page 16

- "About Data and Log Files" on page 16

- "About Snapshot Isolation Levels" on page 17

About the WAN Environment

Applications built on the Ivanti Service and Asset Manager platform are commonly used over a WAN link. Whether deployed in a remote data center or used from branch offices, users expect performance reasonably similar to that of the LAN environment. Following the practices outlined in this guide result in a well performing application over the WAN that closely resembles performance on the local network.

To address poor performance over a WAN, it is important to clearly understand the problem. Though low bandwidth can have an impact, the foremost issue affecting WAN performance is latency, the time it takes data to travel from one endpoint to another. The higher the latency of the WAN link, the longer it takes data to traverse the link. High latency results in delays that can cause users to wait minutes for simple operations to complete.

To combat the higher latency of a WAN environment, Ivanti Service and Asset Manager was designed to make fewer "chunky" calls, rather than frequent smaller "chatty" calls. In other words, instead of retrieving little bits of data in individual requests, the system retrieves larger data sets in a single request. This well-established design principle reduces the overhead of traversing high-latency links, resulting in much improved application performance.

There are certain configurations and features within Ivanti Service and Asset Manager that can affect the behavior of the application, causing it to be more or less chatty, specifically, caching business objects and definitions and using script expressions. This guide covers these areas in detail and provides guidelines for keeping these areas in check.

This section does not describe how to fix network or WAN problems. Rather, this section focuses on ensuring that Ivanti Service and Asset Manager is configured to behave in a manner that produces acceptable performance on a well performing WAN.

Planning and Scheduling Maintenance

Create and follow a regular maintenance plan. Performing regular, ongoing maintenance is crucial to ensuring optimal system performance. At a minimum, a maintenance plan should include the following:

- A schedule for deleting records and tables. See "Deleting Inactive Tables and Records" on page 43.

- Running a tuning advisor tool to determine index tuning actions and then performing those actions. See "Database Tuning" on page 19.

- Scanning for duplicate indexes and then removing the duplicates. See "Checking for Duplicate Indexes" on page 23.

- Configuring database auto-growth. See "Setting the Value for the Database Auto-Growth Parameter" on page 25.

Depending on your system requirements, it is likely that you will perform additional activities as part of a maintenance plan.

The following is an example maintenance plan:

Month	Maintenance Task
January	Evaluate the current archival plan and modify it as necessary.
	Record performance metrics to establish a baseline.
	Run the Microsoft SQL tuning advisor to determine recommended index changes.
	Apply recommended index changes.
	Scan for and delete duplicate indexes.
	Check for license renewal requirements.
February	Evaluate the current archival plan and modify it as necessary.
	Troubleshoot errors reported in the event log and message queuing.
March	Evaluate the current archival plan and modify it as necessary.
	Audit log level and location settings.

	Review database auto-growth and if necessary reset it to an appropriate static value.
April	Evaluate the current archival plan and modify it as necessary.
	Run the Microsoft SQL tuning advisor to determine recommended index changes.
	Apply recommended index changes.
	Scan for and delete duplicate indexes.
May	Evaluate the current archival plan and modify it as necessary.
	Troubleshoot errors reported in the event log and message queuing.
June	Evaluate the current archival plan and modify it as necessary.
	Record performance metrics and compare them with the recorded baseline.
July	Evaluate the current archival plan and modify it as necessary.
	Run the Microsoft SQL tuning advisor to determine recommended index changes.
	Apply recommended index changes.
	Scan for and delete duplicate indexes.
August	Evaluate the current archival plan and modify it as necessary.
	Troubleshoot errors reported in the event log and message queuing.
September	Evaluate the current archival plan and modify it as necessary.
	Review database auto-growth and if necessary reset it to an appropriate static value.
October	Evaluate the current archival plan and modify it as necessary.
	Run the Microsoft SQL tuning advisor to determine recommended index changes.
	Apply recommended index changes.
	Scan for and delete duplicate indexes.
November	Evaluate the current archival plan and modify it as necessary.
	Troubleshoot errors reported in the event log and message queuing.
December	Evaluate the current archival plan and modify it as necessary.
	Record performance metrics and compare them with the recorded baseline.
	Update infrastructure design plans and other documents.
	Update system growth expectations.

About Reviewing, Closing, and Deleting Records

It is not uncommon for systems to have hundreds of thousands of active incident records that no longer need to be open and active. For example, if an incident has not been modified for several months, it can probably be closed.

Dashboard parts and saved searches may query these inactive records daily. Closing records that are older than a specified age (for example, three months) prevents them from being queried by dashboard parts and saved searches, which can significantly improve dashboard and search loading performance. Deleting closed records can also reduce the load on database server infrastructure. Additionally, deleting workflow history and instance tables prevents the workflow engine from having to query them.

We highly recommend that you implement a regularly occurring process to identify records that can be closed and deleted.

You can use service level agreements to automatically trigger a notification when an active record has been open for a specified amount of time. To do this, assign a default service level agreement to all services that do not have an active one. In the service level agreement, set the escalation to notify the owner when an active incident record has been open and active for a certain number of days.

Deleting closed and inactive records can reduce the load on database sever infrastructure. See "Deleting Inactive Tables and Records" on page 43.

Tuning Business Rules and Workflows

The following sections describe how to tune business rules and workflows to maximize performance. Implementing these practices can significantly improve overall system performance and reduce deadlocks and timeouts caused by the workflow engine.

- "Disabling Inactive Business Rules" below

- "Disabling Inactive Workflows" on the next page

- "Deleting Workflow Tables" on the next page

- "Validating Email Addresses in Notification Workflows" on the next page

Disabling Inactive Business Rules

Review all active business rules. If a business rule is no longer used, disable it by removing its trigger and updating the rule description to disabled.

Disabling Inactive Workflows

Whenever a workflow is triggered, the system adds it to the event queue where it waits to be processed. Reducing the number of workflows waiting to be processed in the event queue improves performance. Review all active workflows. If a workflow is not used, disable it and update is description to disabled.

Deleting Workflow Tables

It is not uncommon for workflow history tables to contain hundreds of thousands of records. Workflow instance tables can also contain large numbers of records. Deleting workflow history and instance tables prevents the workflow engine from having to query them. Deleting workflow history and instance tables can significantly improve event table performance.

For details about how to deleting tables, see "Deleting Inactive Tables and Records" on page 43.

We recommend that you delete workflow tables on a regular basis (such as one or more times per week, if feasible).

Validating Email Addresses in Notification Workflows

If a notification workflow containing an invalid email address is triggered, event queue resources are wasted while the invalid email address is evaluated and processed. To prevent this situation, add a condition to notification workflows so that email addresses are validated before the workflow is triggered.

Best Practices for Dashboards and Layouts

You can improve Ivanti Service Manager performance by following the dashboard and layout best practices described in these sections:

- "Dashboard Best Practices" below

- "Layout Best Practices" on the next page

Dashboard Best Practices

Because dashboards are created from multiple dashboard parts, they usually execute multiple saved searches when they are opened or updated. You can significantly improve performance by following these best practices:

- Ensure that dashboards display only active records, and that searches configured in dashboard parts do not search for closed or completed records.

- Disable or increase the duration of automatic refresh on all active dashboard parts.

- Use the fewest possible dashboard parts to create a dashboard (no more than four whenever possible).

- Wherever possible, add date filters and remove the all option.

- Wherever possible, remove memo fields from grids.

- Enable auditing only when necessary.

Layout Best Practices

Ensure that tabs in a workspace exist only if they are absolutely necessary. Workspace tabs that contain counters can consume significant resources. If a tab contains a counter, Ivanti Service Manager executes a query to return the number of records available on that tab. Each tab that generates a query adds to the time it takes for the parent workspace to load.

Improving Back-End Processing

This section describes how to tune stored procedures, triggers, and file import processes to improve performance.

- "Stored Procedures and Triggers" below

- "File Import Processes" below

Stored Procedures and Triggers

We recommend that you configure stored procedures and triggers outside of Ivanti Service Manager to avoid any unnecessary problems.

Stored procedures often consist of complex searches that query multiple database tables. A deadlock can result if a stored procedure is scheduled to run too frequently. Wherever possible, minimize the use of frequently scheduled stored procedures and triggers that query primary business objects such as incident, problem, change, journal, task, and other primary business objects. If possible, do not run these procedures at intervals of less than five minutes. If a frequently scheduled stored procedure is necessary, we recommend that you run it outside of normal business hours.

File Import Processes

If there are too many files to process or import in too short of a time (which might occur if the creation of a new record generates multiple queries), system performance can be significantly impacted.

Wherever possible, schedule the file import process to run outside of normal business hours.

Using Event and Error Log Files for Troubleshooting

Microsoft event logs and message queuing error logs are often helpful in determining common configuration errors, especially with regard to the email listener and file import process. We recommend that you review these logs when you troubleshoot performance issues.

About Using Combo Boxes Sparingly

- "About Using Combo Boxes" below

- "Creating a Pick List" below

About Using Combo Boxes

Do not use combo boxes for validating business objects that have a large number of rows. The maximum number of rows a combo box can display is 200. If there are more than 200 rows, the system does not return the excess rows. If a business object has more than 200 rows, we recommend using a text box with a **Search** dialog box.

To effectively validate business objects with more than 200 rows, perform one of the following:

- Change the business object from a combo box to an associated item selector. Access this function from the field control toolbar.

- Constrain the combo box field to validate using another field to reduce the number of rows returned. For example, constrain the **Employee** field to be constrained by the **Department** field, so that when validating the **Department** field, only employees from the department selected are shown.

Creating a Pick List

1. From the Configuration Console, click **Build > Pick Lists** to open the **Pick Lists** workspace. The system displays a list of pick lists.
2. Click **Add New…**. The system displays the **Add New Validation List** page.
3. Enter information into the fields.

Field	Description
List name	The name of the pick list to be stored.
Display name	The name of the pick list to be displayed in Ivanti Service Manager.
Get list data from	Check **Business Object** and select a business object from the drop-down list.
Display field	The name of the field to display in the pick list.
Value field	A stored field name. Select from the drop-down list.
Sort by field	The field by which to sort. Select a sorting value, either ascending or descending.

4. Add constraints to the pick list to restrict the data presented. Click the **add new** icon to add a constraint formula. This formula is based on the parameter selected in the **Get list data from** field.

5. Click **Save**.

    See the Ivanti Service Manager online help for additional information about creating and modifying a pick list and its values.

Evaluating Slow Running Queries

Queries that take a long time to execute can manifest themselves just about anywhere in Ivanti Service Manager. Dashboards, saved searches, and viewing a business object are all areas impacted by slow running queries. With some familiarity with the tools available, tracking down and fixing these problems is a relatively straight forward task.

- "Identifying Problem Queries" below

- "Improving Saved Search Query Performance" below

Identifying Problem Queries

If queries run slowly, use the Microsoft SQL Server profiling tools to find out which queries are the problem. See "Tuning Indexes" on page 27 for more information.

Improving Saved Search Query Performance

Saved searches are used in many places throughout Ivanti Service Manager, especially in dashboards and workspaces. Saved search queries can be complicated because they can involve related data and typically include criteria for more than just a single field. For this reason, dealing with performance problems for a saved search can be a bit more involved. Constructing saved searches in the most efficient way possible can significantly improve performance.

We recommend the following:

- Minimizing the use of the "Begin with", "Not Begin with", and "Is Not Empty" operators in saved searches.

- Minimizing sorting and grouping in grids.

- Creating index fields for tables frequently used in saved searches to reduce database access time. Use the Microsoft SQL Server profiling tools to find problem queries and to help identify indexes that can improve performance. See the Ivanti Service Manager online help for information about creating indexes.

About Data and Log Files

Microsoft SQL Server databases generally contain two types of files:

- Data files, which usually use the .mdf extension. These files contain the database data.

- Log files, which usually use the .ldf file extension. These files contain log data which can be verbose.

As a best practice, we recommend that you use house these files on different volumes.

About Snapshot Isolation Levels

- "About Snapshot Isolation Levels" below

- "Using Read Committed Snapshot Isolation (RCSI)" below

About Snapshot Isolation Levels

Isolation levels control how two or more transactions running simultaneously are isolated from each other in terms of locking and blocking resources. The isolation level determines the level of concurrency and data consistency. Ivanti Service Manager uses the following isolation levels:

- **Read committed**: This is the default transaction isolation level in Microsoft SQL Server. This isolation level prevents "dirty reads" which occur when a transaction is allowed to read data from a row that was modified by another transaction that has not yet been committed. This level acquires shared locks to prevent other transactions from modifying the data during a read operation by that transaction. A shared lock can be acquired only if there is no exclusive lock (required for data modification) by other transactions; this ensures it reads only committed data.

- **Read committed snapshot isolation (RCSI)**: An extension of read committed but with increased concurrency. With this level, the system uses the last committed version before the statement starts, regardless of when the transaction starts.

We recommend that you use RCSI if your system meets any of the following criteria:

- It has a high number of transactions.

- If you encounter a high number of locks in your database.

- Your system has multiple back-end servers, because there could be multiple workflow, escalation, and other tasks that are triggered after the system creates or updates a record.

Note that using RCSI generally increases the CPU utilization due to the additional processing, but lowers the CPU utilization due to a reduction in locks and deadlocks. The net result is lower CPU utilization and therefore, we recommend using RCSI.

Using Read Committed Snapshot Isolation (RCSI)

For the Ivanti Service Manager database, ensure that the following two parameters are set to **ON**:

- Allow Snapshot Isolation

- Is Read Committed Snapshot On

To set these parameters to on, do the following:

1. Open Microsoft SQL Server Management Studio.

2. Navigate to and highlight the Ivanti Service Manager database to set the properties for.

3. Right-click and select **Properties**.

4. On the left, select **Options**.

5. Under **Miscellaneous**, set **Allow Snapshot Isolation** to **True** and set **Is Read Committed Snapshot On** to **True**.

6. Click **OK**.

Database Tuning

This section describes how to tune the Ivanti Service Manager database for optimal performance. If the system responds slowly while running queries, loading dashboards, or performing other actions, implement the recommendations described here. Information includes:

- "Performance Enhancement Recommendations" below

- "Tuning Indexes" on page 27

- "About Using Full Text Search" on page 34

Performance Enhancement Recommendations

The following sections describe recommendations and caveats to keep in mind when you tune the Ivanti Service Manager database.

- "About Tuning the Indexes on a Regular Basis" below

- "Deleting Inactive Tables and Records" on page 43

- "Checking for Under-Utilized Indexes" on the next page

- "Turning on JSON Compression in Microsoft Internet Information Services (IIS)" on page 21

- "Checking Database Latency and Moving Indexes to a Separate Disk" on page 21

- "About Creating Indexes Using the Configuration Console, Not Third-Party Tools" on page 23

- "Checking for Duplicate Indexes" on page 23

- "System Tables to Not Index" on page 25

- "Not Using Include Fields in Indexes" on page 25

- "Ignoring "Create Statistics" Index Results" on page 25

- "Setting the Value for the Database Auto-Growth Parameter" on page 25

- "Monitoring Database Growth and Resetting the Auto-Growth Parameter as Necessary" on page 26

About Tuning the Indexes on a Regular Basis

Evaluate on an on-going basis the need to perform the re-indexing procedure described in "Tuning Indexes" on page 27. If necessary, you can perform the re-indexing procedure as part of normal maintenance. It does not require hardware or other infrastructure changes.

About Deleting Inactive Tables on a Regular Basis

Whenever possible, delete inactive tables in the Ivanti Service Manager database to keep the database size and complexity to a minimum. See "Deleting Inactive Tables and Records" on page 43 for details about deleting.

Checking for Under-Utilized Indexes

Ivanti Service Manager uses indexes to quickly access data. This is defined as reading from the index. Every time that you add a new record, you must also add the new record information to all of the indexes that it applies to. This is defined as writing to an index. While using an index saves time, adding information (writing) to the index takes time.

We recommend that you check your indexes to ensure that they are used frequently. If an index is not used frequently, the performance impact of writing to it initially might not justify the benefits that it provides. One way to check for under-utilization is to determine the ratio of an index of reads to writes. If you have a table that you are writing records to frequently, but you are not reading using this index very often, it takes more effort to add the new record to the index than it is probably worth.

Calculate the read:write ratio for an index to determine if it is under utilized. If an index has a read:write ratio of 10:1 or less (that is, if total reads to the index does not equal at least 10 times the number of total writes to the index), you should consider deleting the index.

Run the following Microsoft SQL script on the Ivanti Service Manager database to check how often data in an index is read and how often it is written.

```
-- This script checks for under-utilized indexes.
-- Indexes in which the number of "writes" outnumbers
-- the number of "reads" by far, e.g., 90%, are
-- considered possibly under-utilized indexes.

SELECT OBJECT_NAME(s.object_id)AS 'Table Name',

        i.name AS 'Index Name',

        i.index_id,

        user_updates AS 'Total Writes',

        user_seeks + user_scans + user_lookups AS 'Total Reads',

        user_updates - (user_seeks + user_scans + user_lookups)
            AS 'Difference'


FROM sys.dm_db_index_usage_stats AS s WITH (NOLOCK)

        INNER JOIN sys.indexes AS i WITH (NOLOCK)ON s.object_id
            = i.object_id

        AND i.index_id = s.index_id


WHERE OBJECTPROPERTY(s.object_id,'IsUserTable')= 1
```

```
        AND s.database_id =DB_ID()

        AND user_updates > (user_seeks + user_scans + user_lookups)

        AND i.index_id > 1


ORDER BY 'Difference' DESC,

        'Total Writes' DESC,

        'Total Reads' ASC;
```

Turning on JSON Compression in Microsoft Internet Information Services (IIS)

Follow these steps to improve performance by implementing JSON compression:

1. Open the **Computer Management** panel by going to **Start > Control Panel > Administrative Tools > Computer Management**.

2. In the **Computer Management** panel, double-click **Services and Applications** and double-click **Internet Information Services (IIS)**.

3. On the page for the server (not the site) click **Configuration Editor**.

4. Expand **system.webServer** and click **httpCompression**.

5. Click **dynamicTypes** to open it.

6. Click **mimeType** to open the **Properties** dialog box.

7. For the **mimeType** entry, enter **application/json**, and for the **enabled** entry, enter **True**.

8. Click **Apply**.

9. Restart your Microsoft IIS server to load the new module.

Checking Database Latency and Moving Indexes to a Separate Disk

When data and indexes are stored on the same physical disk, resource contention might occur when the system looks up an index and retrieves the corresponding data. Measure the database latency to determine whether resource contention is affecting performance.

Run the following script on the Ivanti Service Manager database to check database latency. If database latency significantly exceeds 110 ms, move indexes to a physical disk that is separate from the disk that stores the Ivanti Service Manager database.

```
-- Description: This script checks the latency on the
-- different Ivanti databases. Ideally the latency figure
-- should be a single digit value.

SELECT

        --virtual file latency

        [ReadLatency] =
```

```
                CASE WHEN [num_of_reads] = 0

                        THEN 0 ELSE ([io_stall_read_ms] / [num_of_reads]) END,

        [WriteLatency] =

                CASE WHEN [num_of_writes] = 0

                        THEN 0 ELSE ([io_stall_write_ms] / [num_of_writes]) END,

        [Latency] =

                CASE WHEN ([num_of_reads] = 0 AND [num_of_writes] = 0)

                        THEN 0 ELSE ([io_stall] / ([num_of_reads] + [num_of_
                        writes])) END,

        --avg bytes per IOP

        [AvgBPerRead] =

                CASE WHEN [num_of_reads] = 0

                        THEN 0 ELSE ([num_of_bytes_read] / [num_of_reads]) END,

        [AvgBPerWrite] =

                CASE WHEN [io_stall_write_ms] = 0

                        THEN 0 ELSE ([num_of_bytes_written] / [num_of_writes]) END,

        [AvgBPerTransfer] =

                CASE WHEN ([num_of_reads] = 0 AND [num_of_writes] = 0)

                        THEN 0 ELSE

                                (([num_of_bytes_read] + [num_of_bytes_
                                written])/

                                ([num_of_reads] + [num_of_writes])) END,

        LEFT ([mf].[physical_name], 2) AS [Drive],

        DB_NAME ([vfs].[database_id]) AS [DB],

        -- [vfs].*,

        [mf].[physical_name]

FROM

        sys.dm_io_virtual_file_stats (NULL,NULL) AS [vfs]

        JOIN sys.master_files AS [mf]

        ON [vfs].[database_id] = [mf].[database_id]

        AND [vfs].[file_id] = [mf].[file_id]


        WHERE DB_NAME ([vfs].[database_id]) like '%HEAT%'
```

```
-- and [vfs].[file_id] = 2 -- log files

-- ORDER BY [Latency] DESC
-- ORDER BY [ReadLatency] DESC
ORDER BY [WriteLatency] DESC;

GO
```

About Creating Indexes Using the Configuration Console, Not Third-Party Tools

Do not use third-party tools (such as Microsoft SQL Server Management Studio) to update indexes in the Ivanti Service Manager database. Instead, use the Configuration Console to create new indexes. We recommend this because the Configuration Console creates and maintains index information as Ivanti Service Manager metadata. Ivanti Service Manager uses this metadata whenever tables or indexes in the Ivanti Service Manager database need to be recreated.

Checking for Duplicate Indexes

Check for duplicate indexes in these situations:

- Before creating indexes in the Ivanti Service Manager database, make sure that the table for which you are creating an index does not already have an index containing the same fields.

- If you suspect that duplicate indexes already exist and are impacting performance, find and delete them.

The following sections describe how to check a specific table to determine whether it is indexed, and how to query the Ivanti Service Manager database to identify all duplicate indexes.

- "Checking a Specific Table for Indexes" below

- "Checking for Duplicate Indexes Using a Microsoft SQL Script" on the next page

Checking a Specific Table for Indexes

Follow these steps:

1. Open Microsoft SQL Server Management Studio.

2. In the **Object Explorer** panel, navigate to the Ivanti Service Manager database.

3. Navigate to **Tables > System Tables >** *table_name* **> Indexes**.

4. Double-click an index to open it in the **Index Properties** window. Indexed fields are listed in the **Index key columns** area.

Checking for Fields that are Already Indexed
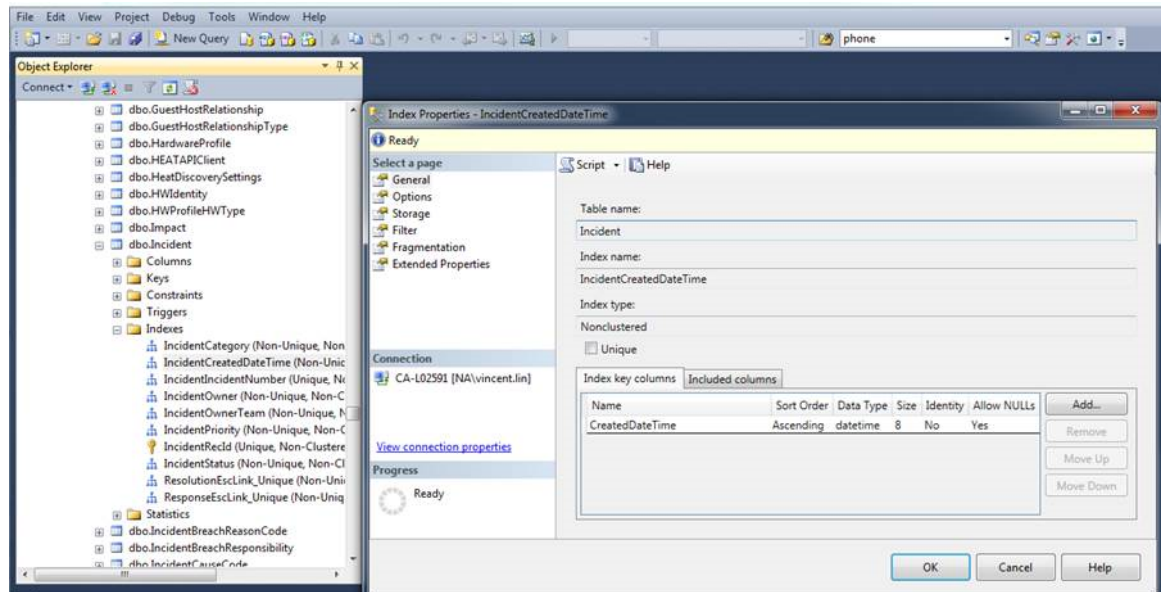
Checking for Duplicate Indexes Using a Microsoft SQL Script

Run the following Microsoft SQL script on the Ivanti Service Manager database to identify duplicate indexes:

```
-- This script checks for duplicate indexes.

with indexcols AS
(
SELECT object_id AS id,

        index_id AS indid, name,

(SELECT case keyno WHEN 0 THEN NULL ELSE colid END AS [data()]

FROM sys.sysindexkeys AS k

        where k.id = i.object_id

        and k.indid = i.index_id

ORDER BY keyno, colid
for xml path('')) AS cols,

(SELECT case keyno when 0 then colid else NULL end AS [data()]
from sys.sysindexkeys AS k

        where k.id = i.object_id

        and k.indid = i.index_id


ORDER BY colid
for xml path ('')) AS inc

FROM sys.indexes AS i
)
```

```
SELECT object_schema_name(c1.id) + '.' + object_name(c1.id) AS 'table',

       c1.name as 'index',

       c2.name as 'exactduplicate'

FROM indexcols as c1

       JOIN indexcols as c2

       ON c1.id = c2.id

       AND c1.indid < c2.indid

       AND c1.cols = c2.cols

       AND c1.inc = c2.inc;
```

System Tables to Not Index

We recommend that you do not index tables that begin with FRS_. These are system tables containing default indexes that are required for Ivanti Service Manager to function properly.

Not Using Include Fields in Indexes

See "Best Practices for Creating New Indexes" on page 32 for more information.

Ignoring "Create Statistics" Index Results

When creating indexes, ignore the "Create Statistics" results returned by performance tuning tools. See "Best Practices for Creating New Indexes" on page 32 for more information.

Setting the Value for the Database Auto-Growth Parameter

By default, the auto-growth size of a new database is 1 MB. This is too small for the Ivanti Service Manager database, so you must increase it.

To estimate and configure the appropriate auto-growth size for your Ivanti Service Manager database, do the following:

1. Estimate the total size of your initial Ivanti Service Manager database based on expected transactions for incident, service request, and other primary business objects.

2. Open Microsoft SQL Server Management Studio.

3. In the **Object Explorer** panel, navigate to the Ivanti Service Manager database.

4. Right click the Ivanti Service Manager database and select **Properties**.

5. In the **Select a page** panel, click **Files**.

6. In the **Initial Size (MB)** column of the Ivanti Service Manager database line item, set the initial database size based on your estimate from step 1.

Specifying the Initial Size of the Ivanti Service Manager Database
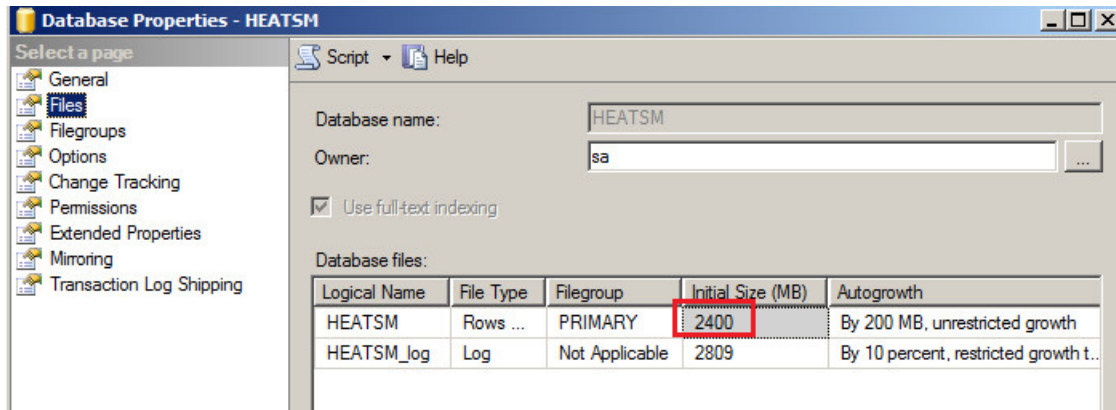
7. Click the **details** icon in the Ivanti Service Manager database line item to open the **Change Autogrowth for Ivanti Service Manager** dialog box.

8. Check **Enable Autogrowth**, select **In Megabytes**, and enter a number that is 5 to 10 percent of the initial database size. We recommend that you set the auto-growth parameter using a static value instead of a percentage.

Setting the Auto-Growth Parameter
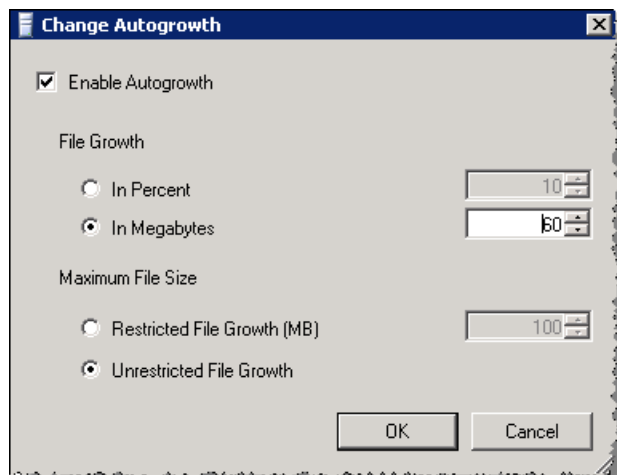
9. Click **OK**.

Monitoring Database Growth and Resetting the Auto-Growth Parameter as Necessary

We recommend that you periodically evaluate whether the value for the **Enable Autogrowth** parameter is appropriate and reset it as necessary.

After you create the initial Ivanti Service Manager database, monitor its size every month for the first few months to estimate its growth rate. If necessary, reset the value for the **Enable Autogrowth** parameter based on the observed growth rate. After you determine the long-term growth pattern, you can usually wait six months or longer before checking and adjusting the value of the **Enable Autogrowth** parameter.

Tuning Indexes

Perform these steps to re-index the Ivanti Service Manager database:

- "Capturing a Database Trace" below

- "Getting Index Tuning Recommendations" on page 29

- "About Creating New Indexes" on page 31

Capturing a Database Trace

Use the Microsoft SQL Server Profiler in Microsoft SQL Server Management Studio to capture a database trace. You analyze the information from this trace file in "Getting Index Tuning Recommendations" on page 29.

1. Open Microsoft SQL Server Management Studio by selecting **Start > All Programs > Microsoft SQL Server 2008 > SQL Server Management Studio**.

2. Highlight the server that hosts the Ivanti Service Manager database.

3. From the **Tools** menu, select **SQL Server Profiler**.

4. Start a new trace:

   - If you are prompted for login credentials, specify them for Ivanti Service Manager and click **Connect**. The system displays the **Trace Properties** window.

   - If you are not prompted for login credentials, select **File > New Trace** in the **SQL Server Profiler** window. The system prompts you for login credentials. Specify credentials for Ivanti Service Manager and click **Connect**. The system displays the **Trace Properties** window.

5. On the **General** tab, do the following:

   a. In the **Trace name** field, enter a name for the trace.

   b. In the **Use the template** field, select **Standard (default)**.

   c. Check **Save to file** and designate a file location for the trace.

Running a Trace From the Trace Properties Window

6. Click **Run** to initiate the trace. The system displays trace information while the trace runs.

A Running Trace

7. When the trace has collected enough information to analyze as described in the next section, select **Pause Trace** or **Stop Trace** from the **File** menu.

Getting Index Tuning Recommendations

Use the Index Tuning Advisor in Microsoft SQL Server Management Studio to get index tuning recommendations.

1. Open Microsoft SQL Server Management Studio by selecting **Start > All Programs > Microsoft SQL Server 2008 > SQL Server Management Studio**.

2. Highlight the server that hosts the Ivanti Service Manager database.

3. From the **Tools** menu, select **Database Engine Tuning Advisor**.

4. Start a new tuning session:

   • If you are prompted for login credentials, specify the credentials for Ivanti Service Manager and click **Connect**. The system displays the **Database Engine Tuning Advisor** window containing a new open session.

   • If you are not prompted for login credentials, select **File > New Session** in the **Database Engine Tuning Advisor** window. The system prompts you for login credentials. Specify credentials for Ivanti Service Manager and click **Connect**. The system opens a new session in the **Database Engine Tuning Advisor** window.

5. On the **General** tab, in the **Workload** area, browse to the trace file that you created in "Capturing a Database Trace" on page 27 and click **Open**.

6. In the **Select databases and tables to tune** area, highlight the Ivanti Service Manager database.

Selecting the Ivanti Service Manager Database
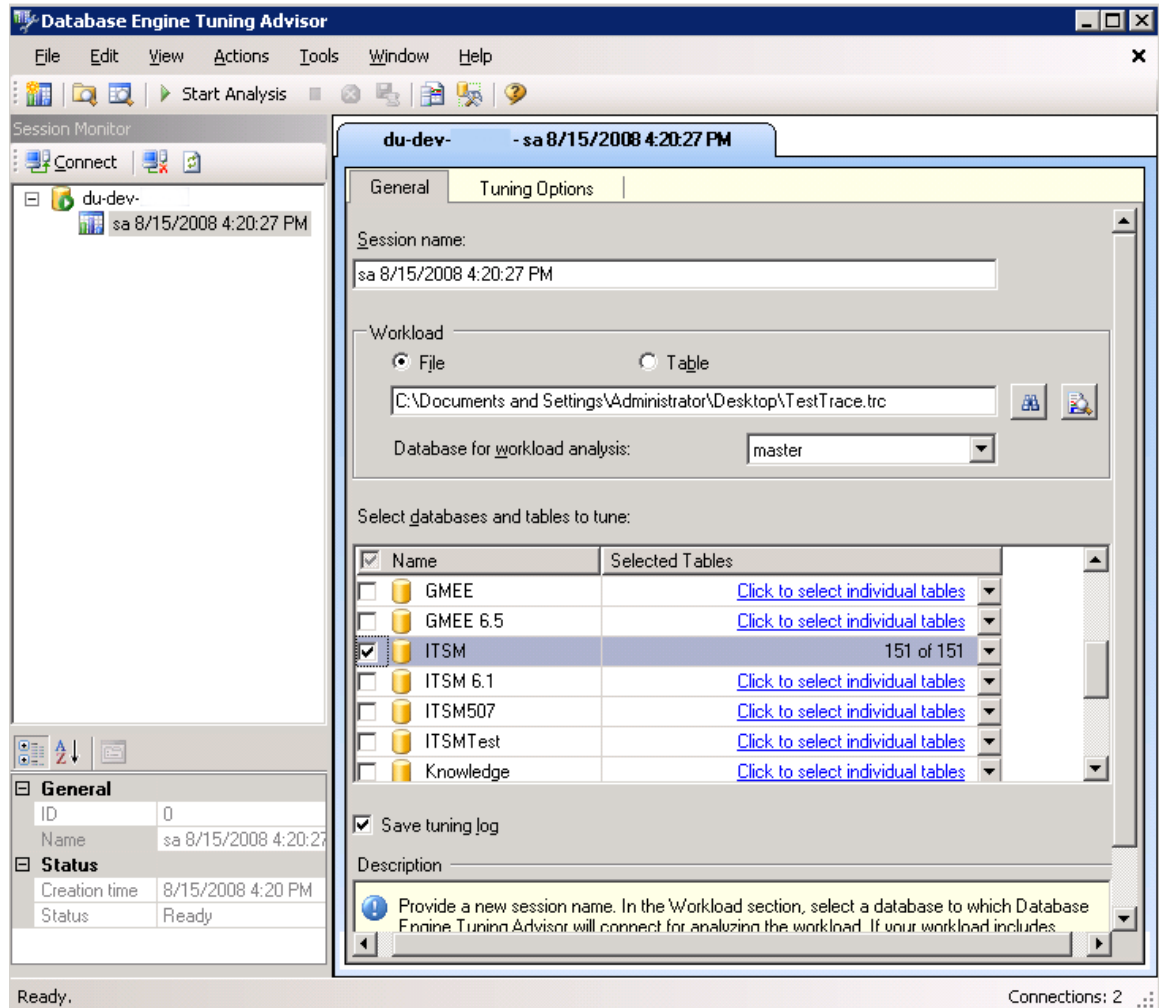
7. From the drop-down menu next to the selected database in the **Selected Tables** column, choose which tables to analyze.

Selecting Tables

8. Click the **Tuning Options** tab.

9. In the **Physical Design Structures (PDS) to use in database** section, select **Indexes**.

10. In the **Partitioning strategy to employ** section, select **No partitioning**.

11. In the **Physical Design Structures (PDS) to keep in database** section, select **Keep all existing PDS**.

12. Click **Start Analysis**.

13. To view progress, select the **Progress** tab.

14. To view recommended indexing actions, click the **Recommendations** tab and go to the **Index Recommendations** section.

Index Recommendations

1. Verify that the indexes recommended for creation do not already exist in the Ivanti Service Manager database as described in "Checking for Duplicate Indexes" on page 23.

2. To view the tuning summary, click the **Reports** tab. You can view reports in the **Tuning Reports** drop-down list beneath the tuning summary.

About Creating New Indexes

After you verify that the indexes recommended for creation do not already exist in the Ivanti Service Manager database, use the Configuration Console to create the indexes.

• "Best Practices for Creating New Indexes" on the next page

• "Creating New Indexes" on the next page

Best Practices for Creating New Indexes

Follow these best practices when you create new indexes:

- Ivanti Service Manager does not create clustered indexes. Disregard any clustered indexes created outside of Ivanti Service Manager.

- Ivanti Service Manager does not create or use include fields or included indexes. You must create and maintain any included indexes that you use outside of the Ivanti Service Manager database. We recommend that you evaluate on a case by case basis the trade off between performance gains and maintenance overhead.

- Ivanti Service Manager maintains statistics only for indexes created in Ivanti Service Manager. Disregard any statistics returned by the create statistics operations in database performance tuning tools.

Creating New Indexes

Perform these steps to create new indexes using the Configuration Console:

1. From the Configuration Console, click **Build > Business Objects**. The system displays the **Business Objects** workspace.

2. Select a business object. The system displays the **Business Object** page.

3. Click the **Indexes** tab. The system displays the **Indexes** workspace with a list of indexes for the business object.

4. To add a new index, click **Add New...**. The system displays index fields associated with this business object.

5. Select the field to index for this business object. You can add more fields later. The system displays the **System Properties** page.

6. Enter information into the fields.

Field	Description
Index Name	A unique name for this index.
Description	A useful description for this index.
Unique	Designates this value as unique, preventing duplicate entries in the index and its backing table. You can create uniqueness by using multiple fields. This option automatically creates an unique index of this field. Unlike the primary key field: You can have multiple unique fields in the business object. This field is allowed to contain NULL values (although the NULL values are not indexed).

Field	Description
Clustered	Checks if the business object is part of a clustered index at the database level. Clustered indexes can greatly increase overall speed of retrieval, but usually only where data is accessed sequentially in the same or reverse order of the clustered index, or when a range of items is selected. This is the same as a clustered index function for a Microsoft SQL database. A clustered index cannot contain included fields (it cannot be a cover index). This checkbox is disabled for index fields that have an increasing value, such as GUID.
Key Fields	See "Viewing, Adding, and Deleting Key Fields" below
Included Fields	See "Viewing, Adding, and Deleting Included Fields " on the next page.
Filter Expression	Optional. You can apply a filter to the index that does not include NULL values. Click the **add new** icon ✚, then choose a field from the list to create a filter. Repeat as needed. **Operation**: Choose an operation from the drop-down list for this filter. **NotNull**: The filter does not index NULL values. **Value**: Enter a string value or expression for this filter.

7. Click **Add this Index**. The system adds the index to the business object.

Viewing, Adding, and Deleting Key Fields

The system displays the running size, in bytes, of the index for the key fields. The limit is 900 bytes.

1. From the Configuration Console, click **Build > Business Objects**. The system displays the **Business Objects** workspace.

2. Select a business object. The system displays the **Business Object** page.

3. Click the **Indexes** tab. The system displays the **Indexes** workspace with a list of indexes for the business object.

4. Click an index name to open the record.

5. To add a key field, do the following:

    a. Click the **add** icon ✚ at the end of the row.

    b. Click **not set** to select a field from the drop-down list.

    c. Click **Ascending** to change the order. You can select either ascending or descending.

6. To delete a key field, click the **delete** icon ✖ at the end of the row.

7. Click **Save**.

Viewing, Adding, and Deleting Included Fields

1. From the Configuration Console, click **Build > Business Objects**. The system displays the **Business Objects** workspace.

2. Select a business object. The **Business Object** page appears.

3. Click the **Indexes** tab. The system displays the **Indexes** workspace with a list of indexes for the business object.

4. Click an index name to open the record.

5. To add a key field, do the following:

    a. Click the **add** icon ✚ at the end of the row.

    b. Click **not set** to select a field from the drop-down list.

    c. Click **Ascending** to change the order. You can select either ascending or descending.

6. To delete a key field, click the **delete** icon ✖ at the end of the row.

7. Click **Save**.

For additional information about adding indexes, see the Ivanti Service Manager online help. (For information about accessing the Ivanti Service Manager documentation, see "Related Documentation" on page 5)

About Using Full Text Search

Full text search support was added to enhance the Ivanti Service Manager database performance. This feature is most helpful when you need to perform similar searches against large text fields such as the **Incident.Symptom** field.

Performance Monitoring

This section describes how to monitor the performance of Ivanti Service Manager.

- "Monitoring the Performance of Servers" below

- "Monitoring the Processing Services" on page 37

Monitoring the Performance of Servers

If you experience performance issues, such as long delays or timeouts, start diagnostics by monitoring the resources used on your system. The following performance monitors provide metrics that you can use to determine if system resources are over utilized.

- On the Ivanti Service Manager application server, monitor the CPU, memory, IO, and network.

- On the Ivanti Service Manager database server, monitor the CPU, memory, IO, and network plus database statistics.

- Also monitor resource utilization. For example, if the average CPU utilization is above 60%, consider adding more CPU on the Ivanti Service Manager application server.

The following are the performance counters on the Ivanti Service Manager application servers. The system samples these counters every five minutes and sends the results:

- **Memory**:

  - Available MBytes

- **Paging file**:

  - Percent usage

- **Physical disk**:

  - Average disk reads per second

  - Average disk writes per second

  - Disk reads per second

  - Disk writes per second

- **Processor**:

  - Percent processor time (CPU)

- **System**:

  - Processor queue length

- Network utilization total, sent, and received per second

The following are the performance counters on the Ivanti Service Manager database server:

- **Memory**:
  - Available MBytes

- **Paging file**:
  - Percent usage

- **Physical disk**:
  - Average disk reads per second
  - Average disk writes per second
  - Disk reads per second
  - Disk writes per second

- **Processor**:
  - Percent of processor time

- **Microsoft SQL Server**:
  - Logins per second
  - Logouts per second
  - Lock waits per second
  - Total latch wait time (ms)
  - Total latch waits per second
  - Buffer manager page life expectancy
  - General statistics – user connections
  - Memory manager – memory grants pending
  - Microsoft SQL statistics – batch requests per second
  - Microsoft SQL statistics – compilations per second
  - Microsoft SQL statistics – recompilations per second

- **System**:
  - Processor queue length

- Average wait time

- Number of deadlocks per second

- Lock timeouts per second

Monitoring the Processing Services

- "About Monitoring the Processing Services" below

- "Workflow Service" below

- "Email Service" on the next page

- "Escalation Service" on page 39

- "Message Queue" on page 40

About Monitoring the Processing Services

You also should monitor the processing services. The Ivanti Service Manager platform includes built-in performance counters for the Ivanti Service Manager message queue, the email service, the escalation service, and the workflow service.

To manage the processing services more effectively, the system implements counters. These counters are implemented per landscape, rather than per service across multiple landscapes. Each of the counters indicates the output across all servers.

For example, the system monitors the counter called "number of tenants" for all servers in the landscape where the service is running. This provides a holistic view of all of the servers and how the service is distributed and consumed. For example, with the workflow service, it shows how the tenants are partitioned versus the non-partitioned servers that act as a catch-all entity.

Workflow Service

The workflow service has these monitors:

Counter	Description
Events handled per poll	Represents the number of events read and handled, per tenant, per poll. If a tenant does not have any events queued, this is zero.
Events handled per second	The number of events read and handled, per tenant, per second.

Counter	Description
Events handled versus read per poll percentage	The number of events handled, versus read, per tenant, per poll percentage.
Events read per poll	The number of events read, per tenant, per poll.
Events read per second	The number of events read, per tenant, per second.
Number of busy processor threads	Based on the configured number of processors of the workflow service on a particular server, in the WorkflowService.exe.config file, the counter shows the number of busy processors available per server. The "number of free processor threads" plus the "number of busy processor threads" should equal the total number of processors in the configuration file.
Number of free processor threads	Based on the configured number of processors of the workflow service on a particular server, in the WorkflowService.exe.config file, the counter shows the number of free processors available per server. The "number of free processor threads" plus the "number of busy processor threads" should equal the total number of processors in the configuration file.
Number of outstanding web service calls	The number of outstanding web service calls.
Number of tenants	The number of tenants assigned to the workflow service, per server. Based on partitioning, the number should indicate the total number of active tenants on the landscape.

Email Service

The email service has these monitors:

Counter	Description
Number of busy mailbox poll threads	Number of busy (consumed) mailbox polling threads, per server. The "number of free mailbox poll threads" plus the "number of busy mailbox poll threads" should equal the total number of mailbox poll threads as configured in the configuration file.
Number of busy message processing threads	Number of busy (consumed) message processing threads available, per server. The "number of free message processing threads" plus the "number of busy message processing threads" should equal the total number of message processing threads as configured in the configuration file.
Number of emails read	The number of emails read from customer mailboxes, per tenant.

Counter	Description
Number of free mailbox poll threads	Number of free mailbox polling threads available, per server. The "number of free mailbox poll threads" plus the "number of busy mailbox poll threads" should equal the total number of mailbox poll threads as configured in the configuration file.
Number of free message processing threads	Number of free message processing threads available, per server. The "number of free message processing threads" plus the "number of busy message processing threads" should equal the total number of message processing threads as configured in the configuration file.
Number of messages read	The number of email messages read from the database, per tenant.
Number of outstanding web service calls	The number of outstanding web service calls.
Number of polled mailboxes	The number of mailboxes polled.
Number of tenants	The number of tenants assigned to the email service, per server. Based on partitioning, the number should indicate the total number of active tenants on the landscape.

Escalation Service

The escalation service has these monitors:

Counter	Description
Events handled per poll	Represents the number of events read and handled, per tenant, per poll. If a tenant does not have any events queued, this is zero.
Events handled per second	The number of events read and handled, per tenant, per second.
Events handled versus read per poll percentage	The number of events handled, versus read, per tenant, per poll percentage.
Events read per poll	The number of events read, per tenant, per poll.

Counter	Description
Events read per second	The number of events read, per tenant, per second.
Number of busy processor threads	Based on the configured number of processors of the escalation service on a particular server, in the EscalationService.exe.config file, the counter shows the number of busy processors available per server. The "number of free processor threads" plus the "number of busy processor threads" should equal the total number of processors in the configuration file.
Number of free processor threads	Based on the configured number of processors of the escalation service on a particular server, in the EscalationService.exe.config file, the counter shows the number of free processors available per server. The "number of free processor threads" plus the "number of busy processor threads" should equal the total number of processors in the configuration file.
Number of outstanding web service calls	The number of outstanding web service calls.
Number of tenants	The number of tenants assigned to the escalation service, per server.

Message Queue

The Ivanti Service Manager message queue has these performance monitors:

Counter	Description
Average message processing time	The average time for a message to be processed by a handler. This does not account for the time that the message spends in the message queue waiting to be dispatched.
Average message size	The average size of the messages received.
Messages created per second	The number of messages received, per second.
Messages processed per second	The number of messages processed, per second by all handlers.
Number of concurrent handlers	The number of handlers that are busy working on items at any given time.
Number of concurrent messages	The number of messages that are being processed (dispatched to its handler) at any given time.

Counter	Description
Number of outstanding web service calls	The number of outstanding web service calls.
Number of tenants	The number of tenants assigned to the message queue server.

Troubleshooting

This section describes how to diagnose and fix common performance issues.

Issues occur occasionally in Ivanti Service Manager. This does not necessarily indicate a malfunction. We recommend that you evaluate each issue on a case by case basis.

Symptom	Possible Cause	Action
Dashboards and searches are slow to load.	Incidents and other records that are no longer active still have open status.	Close and delete inactive incidents as described in "About Reviewing, Closing, and Deleting Records" on page 12 and "Deleting Inactive Tables and Records" on the next page.
Issues and timeouts that are traceable to the workflow engine occur.	Workflows that are inactive are still enabled. Notification workflows do not validate email addresses prior to running.	Disable inactive workflows as described in "Disabling Inactive Workflows" on page 13. Validate email addresses in workflows as described in "Validating Email Addresses in Notification Workflows" on page 13.
Database command timeout errors occur.	You need to re-index the Ivanti Service Manager database. In general, when there is a timeout, the environment has insufficient hardware.	Re-index the Ivanti Service Manager database as described in "Tuning Indexes" on page 27. Verify that your system meets the hardware requirements described in the *System Requirements and Compatibility Matrix for Ivanti Service Manager*.
Workflows take a very long time to execute.		Delete workflow-related tables as described in "Deleting Inactive Tables and Records" on the next page Increase the processor count in the file called **WorkflowService.exe.config** to 40 as described in "Tuning Business Rules and Workflows" on page 12.

Deleting Inactive Tables and Records

This section describes how to delete inactive tables and records in the Ivanti Service Manager database.

- "Best Practices for Deleting Tables and Records" below

- "Deleting Records and Tables from the Ivanti Service Manager Databases" on page 45

Best Practices for Deleting Tables and Records

These sections describe best practices and general recommendations to consider prior to deleting database tables:

- "Identifying the Tables to Delete" below

- "Identifying Criteria for Deleting Tables and Records" on the next page

After you are familiar with these practices and have followed the described recommendations, see the "About Archiving and Deleting Data from the Ivanti Databases" topic in the in the Ivanti Service Manager online help.

Identifying the Tables to Delete

Identifying which tables to delete depends on customer-specific criteria. In general, deleting primary tables (for example, the **Incident**, **Task**, **Journal**, **Delete**, or other tables) results in the biggest performance improvement. Some tables are candidates for deleting in most situations.

Table Name	Primary Key	Foreign Key	Description
Audit_*Business_object_name*		ParentLink_RecID	Multiple tables that contain audit information generated by Ivanti Service Manager. The information is used by Self Service, the workflow engine, and the escalation engine. An example table name is **Audit_Incident**.
Frs_data_escalation_watch		ParentLink_RecID	Contains escalation tracking information.
Frs_data_workflow_instance	RecID	ParentLink_RecID	Contains workflow execution progress information.
Frs_data_workflow_history		WorkflowInstanceLink_RecID	Contains workflow execution history information.

The most commonly deleted transactional data tables are listed in the following table.

Parent business objects have primary keys. Child business objects have foreign keys. Business objects that can be both parent and child have primary and foreign keys.

Table Name	Primary Key	Foreign Key	Description
Attachment	RecID	ParentLink_RecID	Contains attachments displayed in the **Attachments** tab for major business objects (such as incident). Displays and manages all documents or files that are related to a particular business object. Contains the file name, file path, creator's name, and creation date.
FusionAttachments		RecID	A system business object stored inside an attachment containing the actual attachment block.
FRS_MyItem		SecondParentLink_RecID	Contains **My Item** records. The records contain references to incident, service request, and approval requests displayed in the **My Items** tab in Self Service.
Journal		ParentLink_RecID	Contains notes, email messages, and voice activities displayed in the **Activity History** tab for major business objects (such as incident).
Task	RecID	ParentLink_RecID	Contains task information that is displayed in the **Task** tab for major business objects (such as incident). Information pertains to general, computer provisioning, and software installation tasks.

Identifying Criteria for Deleting Tables and Records

You should only delete inactive records. Do not delete active records that contain data that can be referenced.

There are two ways to determine whether a record is inactive:

- **Based on time only**: Select a cutoff date or time, and if the record was not created or modified after the cutoff date or time, consider it inactive. This method is commonly used for records that do not have a lifecycle or different possible statuses (such as the **Audit History** business object). With this method, you specify a date and time to determine if the record is active.

- **Based on time and status**: Select a cutoff date or time and one or more statuses (or lifecycle stages). If the record was not created or modified after the cutoff date or time, or if the record meets the status criteria that you specify, consider it inactive. This method is commonly used for records that have a lifecycle or different possible statuses (such as the **Task** business object). With this method, you specify both a date and time and a status to determine if the record is active.

Deleting Records and Tables from the Ivanti Service Manager Databases

For information about deleting records and tables from the Ivanti Service Manager database, see the "About Archiving and Deleting Data from the Ivanti Databases" topic in the in the Ivanti Service Manager online help.