ptc

integrity™ lifecycle manager

Performance Tuning Guide

11.1

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Contents

1

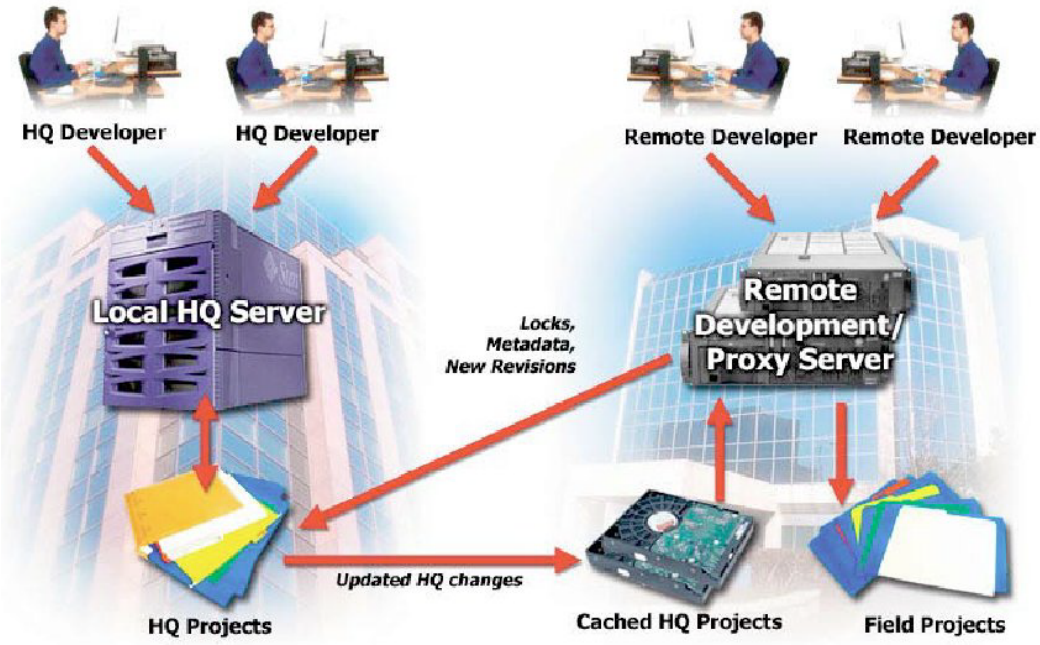
Understanding Resource Usage

This document provides information on how to tune various Integrity Lifecycle Manager components to meet your needs. To provide a better understanding of the product, this section presents requirements information based on the architecture. This information outlines various factors that can cause performance degradation.

When attempting to tune the Integrity Lifecycle Manager server, it is important to recognize that system parameters are not the only factors that affect performance. System hardware must also be sufficient for your needs.

Integrity Lifecycle Manager Architecture

Integrity Lifecycle Manager uses a distributed architecture. The following illustrates this distributed architecture from the perspective of a user.



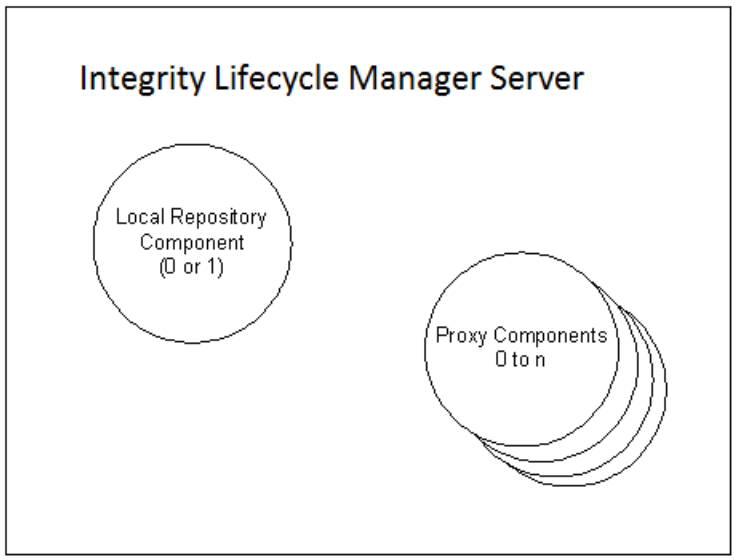
Key Facts:

- Each configuration management project is managed by a unique Integrity Lifecycle Manager server but is accessible from any Integrity Lifecycle Manager client located anywhere in the world. The set of projects (and associated data) managed by an Integrity Lifecycle Manager server is called a *configuration management repository*.

  Similarly each workflows and documents management project (IM) is managed by a unique Integrity Lifecycle Manager server, but is accessible from any Integrity Lifecycle Manager client. The set of types, items, workflow, projects (and associated data) managed by an Integrity Lifecycle Manager server is called a *workflows and documents repository*.
- Integrity Lifecycle Manager servers come in two versions: local and configured with Federated Server™ architecture (FSA). FSA servers are also called proxies. They provide access to the configuration management and workflows and documents repositories of one or more remote servers. These servers can provide access to a local repository as well. A *local server* (also called an *end server*) provides access only to a local repository.

In this document, the *local component* (managing a local repository) is distinguished from the *proxy component* (providing access to a remote repository). This is because a single Integrity Lifecycle Manager server can contain several proxy components, as well as one local component. The requirements of all individual components relate directly to the requirements of the Integrity Lifecycle Manager server. For this reason, it is important to distinguish the notion of a component from the Integrity Lifecycle Manager server itself.

- A single Integrity Lifecycle Manager server can service both configuration management and workflows and documents components, this is frequently the case for proxy components.



- The Integrity Lifecycle Manager server is hosted in an application server (JBoss) and uses a database.
- The Integrity Lifecycle Manager client can access any number of servers. In terms of client performance and memory tuning, it does not matter whether there are proxies between the clients and the end servers.

If your installation does not include FSA, you can ignore the sections that discuss the proxy.

Distributed Caches

Caching is the process of storing copies of data locally for faster retrieval on request. Caching can be particularly valuable when the requesting client and server are geographically distant. Performance is enhanced via a system of distributed caches. Integrity Lifecycle Manager servers and proxies have shared

caches so that users can benefit from each other's work. Integrity Lifecycle Manager clients also have personal caches containing only the data of interest to the user.

Configuration Management

- The configuration management *bulk data cache* (BDC) accelerates the retrieval of frequently accessed revisions contents (referred to as bulk data). BDCs use both memory and disk space, and their contents are preserved across reboots.

  Configuration management BDCs exist only on the Integrity Lifecycle Manager server (or proxy). There is no configuration management BDC on the Integrity Lifecycle Manager client.
- The *metadata cache* contains data on project structure, locks, labels, and other information. Metadata caches are fully managed in memory. Their contents are saved to disk to remain populated after a reboot.

  There are metadata caches on both the Integrity Lifecycle Manager server and the Integrity Lifecycle Manager client.

Workflows and Documents

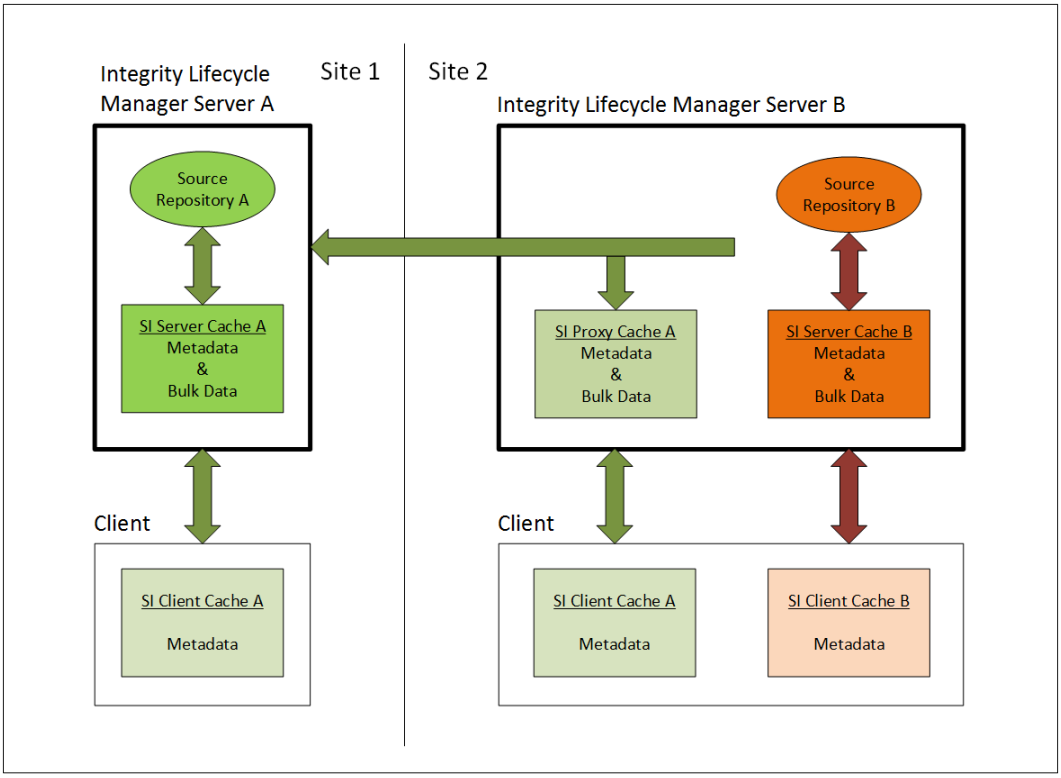
- The workflows and documents BDC accelerates the retrieval of attachments. BDCs use both memory and disk space, and their contents are preserved across reboots. Workflows and documents BDCs exist on the Integrity Lifecycle Manager server (or proxy) and clients. The client BDC is only used when dealing with attachments in rich-text fields such as pictures, as well as when retrieving pictures from item presentation templates.
- A small workflows and documents metadata cache is present in each component and currently only used for permission checks and data integrity.

It is important to remember that each cache manages data for a single repository. This could be either a configuration management repository or a workflows and documents repository. As a result, the Integrity Lifecycle Manager server manages one set of caches per repository and the Integrity Lifecycle Manager client also manages one metadata cache per repository. When accessing multiple repositories, cache resources, such as disk space, add up.

In the following configuration management illustration:

- Two configuration management (SI) repositories are involved—one hosted by Server A and the other by Server B. Server B is also a proxy for Repository A. It is intended to improve access to Server A, which is at a different site and connected by a slow network.
- In each entity, the configuration management (SI) caches provide access to a single repository. There are server caches on the server that own the

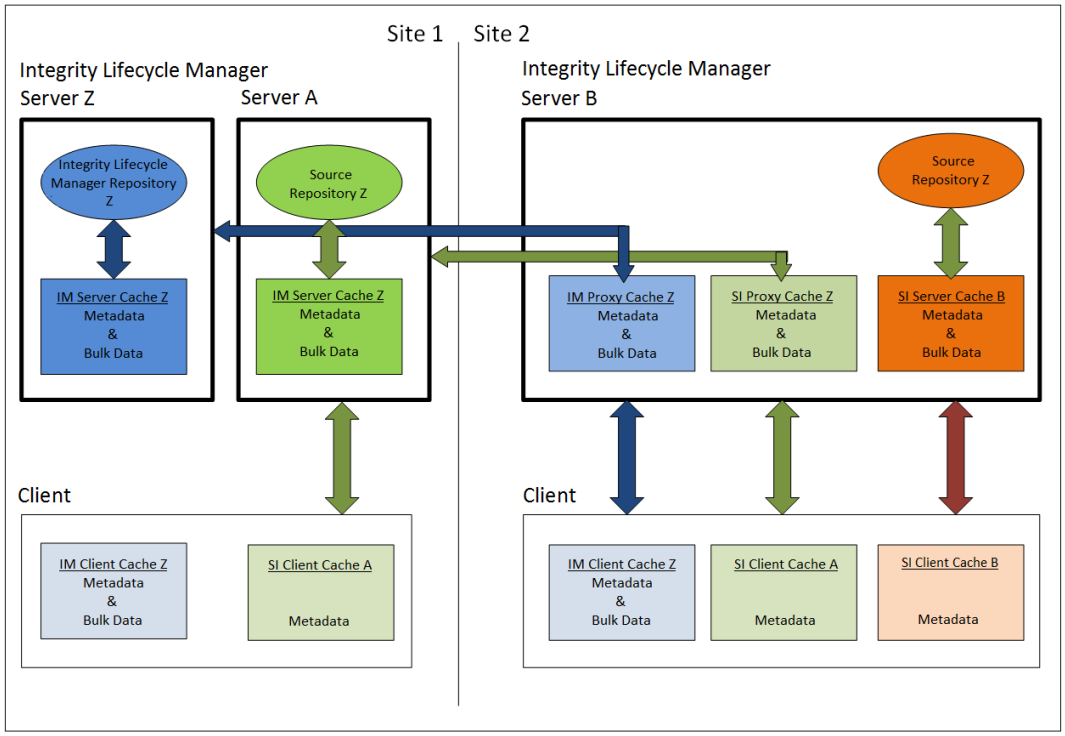
repository, proxy caches on site 2, and finally client caches. Both the server caches and proxy caches contain metadata and bulk data, while client caches contain only metadata.



Configuration management caches also provide real-time notification of project changes. The propagation of events from the Integrity Lifecycle Manager server to the Integrity Lifecycle Manager client can use significant amounts of temporary memory. This temporary memory is on the local server, any proxies, and the Integrity Lifecycle Manager client where they are being processed.

In the following mixed configuration management/workflows and documents illustration:

- The same configuration management (SI) repositories are involved. A workflows and documents repository (Integrity Lifecycle Manager Repository Z) was created on a separate Integrity Lifecycle Manager server on site 1. The unique Integrity Lifecycle Manager server on site 2 also acts as a workflows and documents proxy for Repository Z.
- In each entity, new caches provide access to the additional workflows and documents repository. There are server caches on the server that own the workflows and documents repository, proxy caches on site 2, and finally client caches. Each of these caches contains both metadata and bulk data.



Once again there is one set of caches for each application and for each connection.

Java Memory Usage

Integrity Lifecycle Manager software is based on Java technologies. This section outlines the memory profile of a Java application.

Unlike programs written in traditional languages (such as C), Java programs are not compiled to native machine code but to an intermediate language. This intermediate language is called byte-code and is run inside the Java Virtual Machine (JVM). The memory usage reported by the Task Manager (in Windows), or by `ps` (in UNIX), includes application requirements and memory used by the JVM.

The memory used by the JVM process can be divided into the following areas:

- Heap

  This is the area where the *Garbage Collector* (GC) manages the Java application data. All threads share the heap. Minimum and maximum sizes of the heap can be customized. The maximum size parameter is the most important JVM parameter, because when the heap memory is exhausted, the application throws out-of-memory errors.

- JIT Compiled Code

Java applications are not statically compiled to native machine code. To achieve good performance, most JVMs use a just-in-time compiler (JIT) that compiles and stores frequently called operations.

Note

The Integrity Lifecycle Manager server uses the HotSpot server JIT, while the Integrity Lifecycle Manager client uses the HotSpot client JIT compiler. Both use a default area of up to 32 MB for compiled code.

- Stack Area

  Java applications are highly multi-threaded. Each thread uses its own stack to store data, such as method calls with parameters and return values or local variables.

Note

The Integrity Lifecycle Manager server assigns 128 KB of stack memory to each thread while the client assigns 256 KB to each thread.

Java applications can also communicate with other native applications. Those processes have their own memory needs. It is important to understand that a lack of either heap or system memory results in poor performance or failure. Setting higher heap sizes can also affect performance. For additional details on detecting and adjusting memory settings, see Configuring and Tuning the Integrity Lifecycle Manager server on page 29.

Java Garbage Collection Overview

Java applications benefit from JVM automatic memory management where objects in the heap are allocated and cleaned-up at runtime. From time to time, the GC runs to reclaim unused objects.

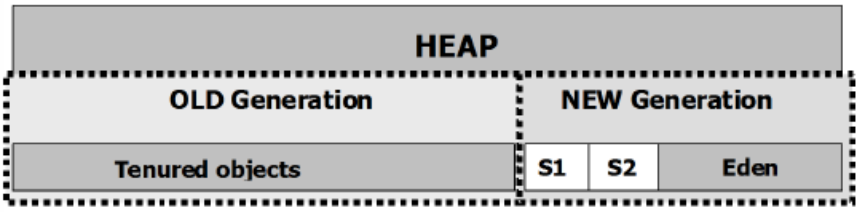
In certain situations, the garbage collection can hamper application execution by causing long or unnecessary pauses. When this happens, it is sometimes possible to reduce the overhead of the GC by tuning certain parameters.

Garbage collection techniques used by the JVM are subject to change. Therefore, information contained in this document is only valid for the HotSpot version used by configuration management (for your release of Integrity Lifecycle Manager). Configuration management ships with CMS and falls back to stop-the-world generational GC. A stop-the-world generational GC requires all threads to stop working due to its execution.

By default, concurrent mark sweep (CMS) garbage collection is used. CMS GC attempts to mark instances for removal, with most phases concurrent with application execution.

Tuning the CMS consists of manipulating parameters to avoid the need for a full stop-the-world GC.

In a generational GC, the heap is divided into a new generation and an old generation, as shown in the following illustration.



Most objects are short lived and significant memory is reclaimed by collecting the new generation only. Non-generational GCs have to browse the entire heap, resulting in long pauses.

New objects are allocated within “Eden”. When Eden becomes full, a minor collection is triggered in the new generation. Hopefully, few objects are still alive (that is, in use). Any live objects are moved to a survivor area in S1 and S2 in the new generation. Objects that survive several minor collections become tenured and are then moved to the old generation.

When the old generation area becomes full, a major collection is triggered. This is typically a slow operation because it involves all living objects.

Tuning the GC involves finding a balance between the sizes of all the relevant areas to minimize GC side effects. These side effects can include pauses, delayed collections, and slowdowns.

For more information on GC parameters, see Configuring and Tuning the Integrity Lifecycle Manager server on page 29.

2

Integrity Lifecycle Manager Entities

This document provides information on how to tune various Integrity Lifecycle Manager components to meet your needs.

To provide a better understanding of the product, this section presents information about the Integrity Lifecycle Manager server, the Integrity Lifecycle Manager proxy server, and the Integrity Lifecycle Manager client.

This information outlines various factors that can cause performance degradation.

Integrity Lifecycle Manager server With Local Configuration Management Repository

An Integrity Lifecycle Manager server with a local configuration management repository uses different types of resources including, CPU, memory, disk, and database resources. To help with understanding resource consumption, this section provides details on the major components of the Integrity Lifecycle Manager server.

All source repository data is stored in the database, which includes (but is not limited to) project and archive information. This includes archive file contents.

Local Component (Managing the Local Configuration Management Repository)

- Repository (Database + Disk Space in Gigabytes)

  The repository contains all versions of project and archive data. The repository grows over time as new versions of files and projects are saved. The database hosting the repository must be on a drive with sufficient available drive space capable of high disk throughput speeds.
- Additional configuration management data (Memory + Database)
  - Project registry and locks storage are contained in a database table.
  - ACLs—controlling all operations against the local repository—are also stored in a database table.
- Performance
  - SI Server Metadata Cache (Memory + Database)

    This is an in-memory cache providing very fast access to the most frequently used metadata in a configuration management project. The amount of memory used by the server metadata cache depends on the number, size, and age of the projects. The root cache maximum memory is customizable. For more information, see `si.ServerCache.default.size` in Adjusting Integrity Lifecycle Manager server IM Cache Resources on page 45.

    The root cache also uses database resources to store a journal of recent events. This allows the client and proxy caches to recover after any communication disruption. The database resources used by the cache journal are customizable. For more information, see `si.ServerCache.default.journal` (Low/HighWater) in Adjusting si.Cache.maxServerPoolSize on page 45.

    Real-time event propagation requires large amounts of temporary memory.

- SI BulkData Cache (Memory + Disk)

  The BulkData cache accelerates the retrieval of frequently accessed revision contents referred to as bulk data. These revisions are stored on disk and mapped in memory. For more information, see `si.ServerCache.default.bulkMemSize` and `si.ServerCache.default.bulkDiskSize` in Adjusting si.Cache.maxServerPoolSize on page 45.

- Triggers (CPU + Memory)

  Because triggers can be designed to do almost anything, it is important to take their requirements into account when estimating server requirements.

- Other Services
  - Authentication (Memory)

    The authentication service controls access to the Integrity Lifecycle Manager server. Resource requirements depend on the type of authentication realm, and, to a certain extent, on the number of groups and users.

  - Logging (Memory + Disk)

    Selected debugging levels increase memory and CPU usage, and result in more messages being written to the log files. Configuration is also available to specify how many log files to keep, and how large these files can grow to in size.

Integrity Lifecycle Manager server With Local Workflows and Documents Repository

An Integrity Lifecycle Manager server with a local workflows and documents repository uses similar resources as an Integrity Lifecycle Manager server with a local configuration management repository as all products use a common architecture.

Managing the Local Workflows and Documents Repository

- Repository (Database + Disk Space in Gigabytes)

  The workflows and documents database contains all workflows and documents data such as items, queries, or attachments. The repository grows over time, with new items and time entries created daily. The workflows and documents database must be given plenty of free space.

- Users accessing this repository (Memory + CPU)

The total number of users and activity affects resource requirements.

- Performance
  - IM bulk data cache (Memory + Disk)
  - The bulk data cache accelerates the retrieval of frequently accessed attachment contents referred to as IM bulk data. These attachments are stored on disk and mapped in memory. For more information, see `im.ServerCache.default.bulkMemSize` and `im.ServerCache.default.bulkDiskSize` in Adjusting Integrity Lifecycle Manager server IM Cache Resources on page 45.
- IM server metadata cache (Memory)

  A small IM metadata cache is present in each component and currently only used as a helper to the IM bulk data cache. The amount of memory used by the IM server metadata cache should stay relatively small and not require any adjustments to the defaults.
- Other Services (Memory + Database)
  - Authentication (Memory)

    Controls access to the Integrity Lifecycle Manager server. Resource requirements depend on the type of authentication realm, and, to a certain extent, the number of groups and users.
  - ACLs—controlling all operations against the local repository—are also stored in a database table.
  - Logging (Memory + Disk Space)

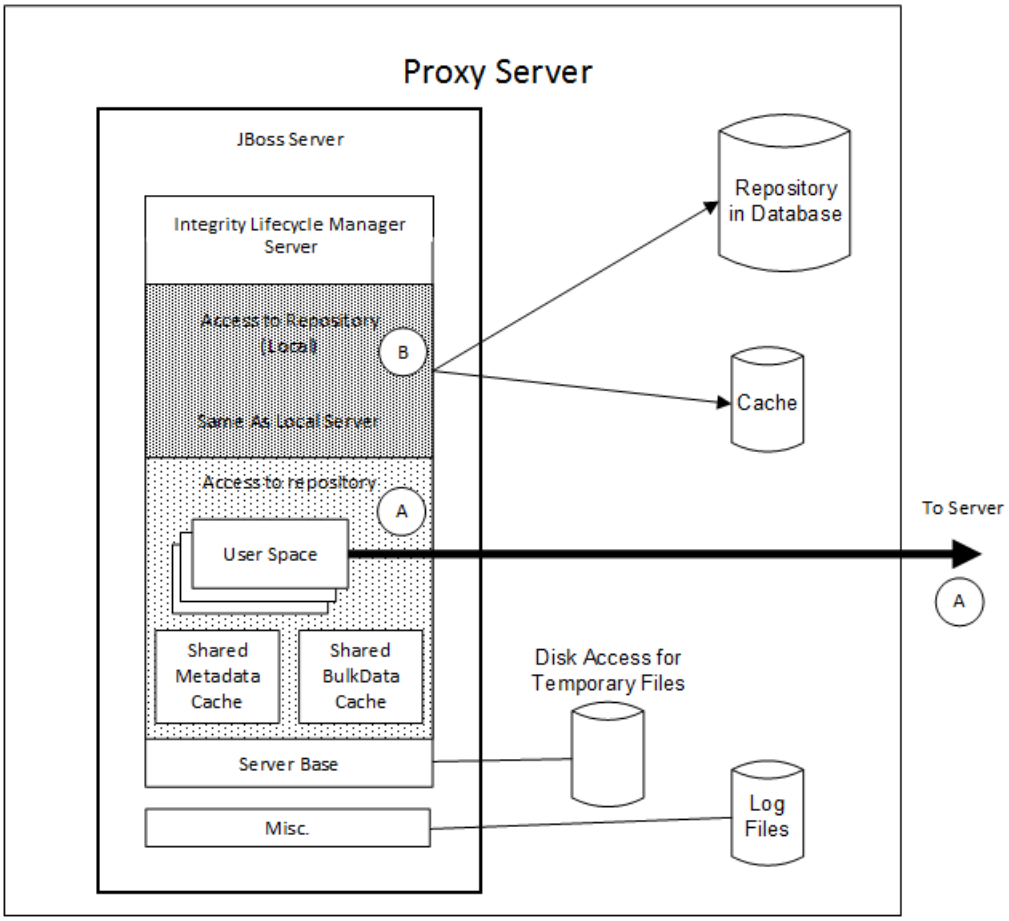
    Selected debugging levels increase memory and CPU usage, and result in more messages being written to the log files.

> **Note**
>
> The Integrity Lifecycle Manager server can also host configuration management. Configuration management requirements should be taken into account when using a single Integrity Lifecycle Manager server for both applications. In particular, there is one set of caches for each application.

Integrity Lifecycle Manager Proxy Server

This section focuses on the Integrity Lifecycle Manager server when enabled for FSA, and provides a summary of the main parts of an Integrity Lifecycle Manager server with proxy components. The purpose of this section is to provide an understanding of the resource requirements for the FSA configuration.

The following illustrates the Integrity Lifecycle Manager server with one local component and one proxy component. It applies to both workflows and documents and configuration management. The Integrity Lifecycle Manager server application is displayed on the left, with disk and database resources displayed on the right.



Managing a Local Repository

In the FSA configuration, the local component of the proxy is the same as for the local server. For details, see the previous section entitled Integrity Lifecycle Manager server With Local Configuration Management Repository on page 16.

Each Proxy Component (Accessing a Remote Repository)

Note

The total number of proxied servers directly affects resource requirements. When a single Integrity Lifecycle Manager server is used with configuration management and with workflows and documents, it counts as two servers resource-wise on the proxy. In particular, there is one set of caches for each application.

- IM repository and ACLs

  The repository is remote and does not use any local disk or database resources with the exception of attachments. See the following subsection entitled "Performance."
- Users accessing the proxied repository (Heap)

  Each user connected through the Integrity Lifecycle Manager client uses a portion of the proxy heap (the user space). The total number of users affects resource requirements. Only those users operating an Integrity Lifecycle Manager client can access the remote repository through a proxy component.
- Performance

  Because the repository is remote, the performance module becomes more important and is usually allocated more resources than the corresponding root caches.

  - IM proxy bulk data cache

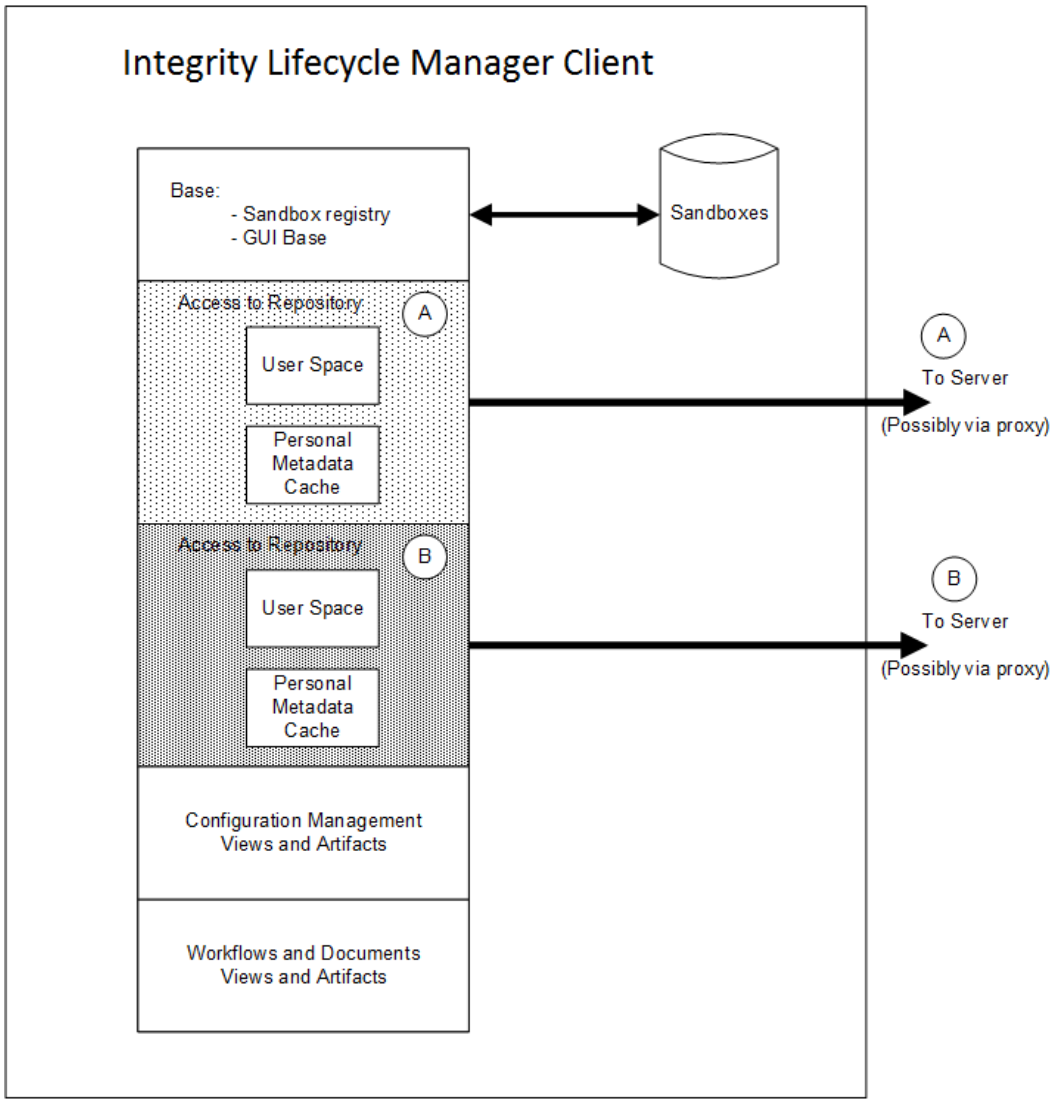
    The bulk data cache accelerates the retrieval of frequently accessed attachment contents referred to as IM bulk data. These attachments are stored on disk and mapped in memory. Because of the time it takes to download files over a low-bandwidth network, this is one of the most important elements to tune in a proxy. For more information, see `im.ProxyCache.bulkMemSize` and `im.ProxyCache.bulkDiskSize` in Adjusting Proxy Settings on page 52.
  - IM proxy metadata cache

    A small IM metadata cache is present in each component and currently only used as a helper to the IM Bulk Data Cache. The amount of memory used by the IM server metadata cache should stay relatively small and not require any adjustments to the defaults.
- Other services

Authentication and logging services are the same as for the local server. For more information, see the previous section for Integrity Lifecycle Manager server With Local Configuration Management Repository on page 16 or Integrity Lifecycle Manager server With Local Workflows and Documents Repository on page 17.

Integrity Lifecycle Manager client

This section focuses on the Integrity Lifecycle Manager client entity. The following illustrates general components and requirements of the Integrity Lifecycle Manager client. It applies to both workflows and documents and configuration management. The left side shows the Integrity Lifecycle Manager client running in memory, while the right side shows disk resources.



Access to a Remote Repository

The total number of server connections directly affects resource requirements.

Note

When a single Integrity Lifecycle Manager server is used for configuration management and for workflows and documents, it counts as two servers resource-wise on the client: in particular there is one set of caches for each application.

Other Client Services

- Communication with the Integrity Lifecycle Manager server
- Logging

Note

The preceding information also applies to IDE integration users. Web users can ignore this section and refer to Integrity Lifecycle Manager server With Local Configuration Management Repository on page 16.

Configuration Management

- Manipulation of the remote configuration management repository (Heap)

  This uses part of the Integrity Lifecycle Manager client heap. The percentage size varies depending on the number of objects with which you are actively working.
- Configuration management performance

  The client personal metadata cache (Memory) is an in-memory cache that accelerates access to metadata. The amount of memory it uses depends on the number, size, and age of your projects. The maximum memory of the client cache is customizable.

  Real-time event propagation requires large amounts of temporary memory.
- Sandboxes
  - Sandbox working files (Disk), as affected by the total number of sandboxes.

The number and size of files in the corresponding project directly affects the amount of disk space used by a sandbox.

- Registry

  The client registry stores sandbox information, working file versions, and information on any deferred operations the user may have.

- Configuration management user interface (Memory)

  All graphical views require large amounts of memory including:

  - Sandbox and history views
  - The three-way differencing or merge utility
  - Permission management panels

  Command-line views require small amounts of memory.

Workflows and Documents

- Working with the remote IM repository (Memory)

  This uses part of the Integrity Lifecycle Manager client memory. The percentage size varies depending on the number of objects with which you are actively working.

- Integrity Lifecycle Manager performance

  - IM personal bulk data cache

    The client bulk data cache accelerates the retrieval of rich text attachment content while regular attachments are directly retrieved from the server (or proxy). This accelerated retrieval also takes place when retrieving pictures from item presentation templates. These attachments are stored on disk and mapped in memory.

    For more information, see `im.ClientCache.bulkMemSize` and `im.ClientCache.bulkDiskSize` in Adjusting Proxy Settings on page 52.

  - IM proxy metadata cache (Memory + Disk)

    A small IM metadata cache is present in each component and currently only used as a helper to the IM Bulk Data Cache. The amount of memory used by the IM server metadata cache should stay relatively small and not require any adjustments to the defaults.

  - Integrity Lifecycle Manager user interface (Memory)

    - If you use either the Integrity Lifecycle Manager command line or graphical user interfaces, be aware that all Integrity Lifecycle Manager applications share client resources.

- The Integrity Lifecycle Manager web interface does not use Integrity Lifecycle Manager client resources (besides the web browser), but those of the Integrity Lifecycle Manager server.

3

Configuring and Tuning

This section provides instruction and information about setting up Integrity Lifecycle Manager for the best performance. It also covers reassessment and adjustment if suboptimal performance develops.

Before Configuring and Tuning

Ensure the following:

- Available hardware is sufficient for the system requirements.
- Recommended configurations are applied.
- All performance-related upgrades (such as patches or service packs) are applied.

Tuning Methodology

The following general recommendations apply:

- Begin with default settings.
- Observe the system for any indication of suboptimal behavior.
- Use diags or review log files to confirm that the new settings have been applied properly.
- Observe again until the correct tuning is obtained.

Requirements will evolve over time as projects grow and more users access the server. Additional clients and proxies may also access additional local servers.

PTC recommends that you periodically check for signs of performance degradation and adjust settings if necessary.

Performance Analysis Tools

In addition to the analysis of log files, certain tools can be used to help with performance tuning.

Stats

This command provides detailed statistics about commands, caches, triggers, and more, for both configuration management and workflows and documents. This is, together with log file analysis, the first thing to look at when dealing with potential performance issues.

Usage:

```
si stats –-target=server or si stats –-target=proxy or
si stats --target=client
im stats –-target=server or im stats –-target=proxy or
im stats --target=client
```

Diags and Cache Diags

Note

The `si diag` and `im diag` commands are not supported for general customer use. Only use the `si diag` and `im diag` commands under the guidance of a representative from PTC Technical Support.

- `cacheInfo`

  This diag reports on metadata, bulkdata, bulkdata caches, and heap usage.

  Usage:

  ```
  si diag –-diag=cacheInfo --target=[server|client|proxy]
  im diag –-diag=cacheInfo --target=[server|client|proxy]
  ```

- `cacheSettings`

  This diag reports on current cache settings for a particular configuration management or workflows and documents connection.

  Usage:

  ```
  si diag –-diag=cacheSettings --target=[server|client|proxy]
  im diag –-diag=cacheSettings --target=[server|client|proxy]
  ```

- `Support package`

  Support packages now contain status information about all caches IM and SI running in the specified target server or client. This is especially useful on a proxy to quickly see the total resources used or allocated to all caches.

  Usage:

  ```
  si diag --diag=collectSupportPackage --output=supportPackage.zip
  ```

  `Server.log` contains the equivalent of the `CacheInfo` diag for all caches, IM and SI.

- `Connections`

  This diag reports on current connections for both configuration management and workflows and documents.

  Usage:

  ```
  im diag –-diag=connections –-target=[server|proxy]
  ```

- `setSizes` Cache Diag (IM only)

  Dynamically change cache sizes.

  Can be used during the tuning process to change cache sizes without restarting the server. This diag does not persist new values: properties files must be updated once tuning is finished.

Examples:

- Provide total disk space, everything else computed dynamically

  ```
  im ci.CacheDiag --name=setSizes --param=1G --target=proxy
  ```

  Will set: BDC: 1G/8Mb MC: 8M
- Provide total disk and memory space, sized computed dynamically

  ```
  im ci.CacheDiag --name=setSizes --param=500M --param=25M --target=proxy
  ```

  Will set: BDC: 500G/12.5Mb MC: 12.5M

- `viewallobjectsforsession`

  This diag allows the administrator to bypass all sharing permissions and view Integrity Lifecycle Manager objects (queries, reports, charts, dashboards) for all users in a current session. This is useful when debugging and correcting poorly performing business processes without requiring end users to share objects or send information outside the system.

  Note

  This setting is cleared when the session ends by disconnect or by exit.

  Usage:

  ```
  im diag --diag=viewallobjectsforsession [on|off]
  ```

Process and System Metrics

Each operating system includes a set of system tools that can be used to show process, CPU, and memory information. TaskManager—provided on all professional versions of Windows—is such a tool. PerfMeter and ps are the equivalent tools for UNIX.

4

Configuring and Tuning the Integrity Lifecycle Manager server

This section focuses on end servers (that is, Integrity Lifecycle Manager server that have FSA disabled and are not proxied to another server). If you are tuning a proxy, also read the section entitled Configuring and Tuning the Proxy on page 51

Identifying Server Performance and Memory Issues

This section discusses how log file analysis and server checks can be used to optimize the performance of the Integrity Lifecycle Manager server.

Location of Server Log Files

Server log files are located in the log subdirectory under the Integrity Lifecycle Manager server install directory. The active log file is called `server.log` and previous log files have a sequential, numbered suffix as indicated in the following examples for UNIX and Windows:

UNIX (Solaris, Linux, HP-UX):

install dir`/log/server.log`

install dir`/log/server.log.xx`

Windows:

install dir`\log\server.log`

install dir`\log\server.log.xx`

Server Out of Java Heap Memory

When the server begins to run out of heap memory, the Java garbage collector runs more frequently. This causes the application to slow down or may be temporarily unresponsive.

When the server is completely out of heap memory, messages such as the following are generated:

```
Tue Sep 24 11:44:04 EDT 2002:<E> <Adapter>
OutOfMemoryError in Adapterjava.lang.OutOfMemoryError
<<no stack trace available>>
```

Or:

```
* * ERROR * * * * (5): Not enough memory:
The server application ran out of memory on myServerName.
at mks.frame.cache.Cache.<clinit>(Cache.java:70)
at mks.si.brain.cache.CacheGlue.init(CacheGlue.java:193)
at mks.si.brain.cache.CacheGlue.initCacheGlue(CacheGlue.java:234)
[...]
```

These messages are generally propagated to the Integrity Lifecycle Manager client. They are displayed in error windows, and they are written to the `server.log` on the server.

Error messages of this type mean that the Java heap requirements are greater than the current settings. To resolve this problem, increase the server maximum heap size.

Key points:

- A common mistake is to assume that the second error message is indicating that cache resources are too small because the stack trace refers to cache. “Out of memory” means “out of heap,” regardless of the circumstances. Allocating more resources to the cache does not correct the problem because the server then uses even more memory. When a cache is too small, a specific message is displayed. For more information, see Integrity Lifecycle Manager server Caches Too Small on page 33.
- Given the distributed architecture of configuration management, out-of-memory errors for the Integrity Lifecycle Manager client are sometimes logged on the server. If the message contains information similar to the following, it means that the Integrity Lifecycle Manager client is running out of memory. No adjustment is required for the Integrity Lifecycle Manager server:
  ```
  * ERROR * * * * (5): The Integrity Client is out of memory.
  Contact the Integrity Administrator for the corrective action.
  ```
- When running FSA, several Integrity Lifecycle Manager servers might be running in different locations. To determine which server has run out of memory, you must check all messages carefully. Contact PTC Technical Support if you are uncertain which server has run out of memory.

Garbage Collector Slowing Down the Integrity Lifecycle Manager server

If you experience disruptive pauses or the Integrity Lifecycle Manager server becomes slow, it is possible that the garbage collector (GC) is not performing at its best. To check GC performance, you must enable JVM GC statistic logging. Take the following steps:

1. Enable verbose garbage collection

   The first step is to enable a verbose garbage collection based on the operating system of the Integrity Lifecycle Manager server.

   - On Windows

     To make the required modifications for verbose garbage collection on Windows, do the following:

     - Edit *`install dir`*`\config\mksservice.conf`
     - Add `-verbose:gc` and `-Xloggc:gc.log` to `mks.java.additional` list of arguments.

       For example:
       ```
       mks.java.additional.23=-verbose:gc
       ```

```
mks.java.additional.24=-Xloggc:gc.log
```

Ensure that the numbers trailing the `mks.java.additional` property are unique and not repeated anywhere in the `mksservice.conf` file.

- Restart the service. GC data is sent to *`install dir`*`\bin\gc.log`.

- On UNIX

  To make the required modifications for verbose garbage collection on UNIX, do the following:

  - Edit *`install dir`*`/config/mksservice.conf`.
  - Add `-verbose:gc` and `-Xloggc:gc.log` to `mks.java.additional` list of arguments. For example:
    ```
    mks.java.additional.23=-verbose:gc
    mks.java.additional.24=-Xloggc:gc.log
    ```

    Ensure that the numbers trailing the `mks.java.additional` property are unique and not repeated anywhere in the `mksservice.conf` file.
  - Stop and restart the Integrity Lifecycle Manager server. GC data is sent to *`install dir`*`\bin\gc.log`.

Note

The new settings take effect immediately after restarting the server.

Also be aware that the `verboseGC` option does slow down server execution. Therefore, you should remove the `verboseGC` option and restart the server as soon as you have completed your analysis.

2. Read the trace

   The next step is to read the trace, for example:
   ```
   [GC 123402->43019K(574762K), 0.1347871 secs]
   [GC 125715K->41278K(574762K), 0.1447228 secs]
   [Full GC 150456K->46835K(574762K), 0.997724 secs]
   ```

   - The first two lines show minor garbage collections, while the third line shows a major collection.
   - On each line, the last number is the time taken by each collection.

- The first two numbers represent the cumulative size of objects, possibly in use before and after the collection has occurred.
- The number in parentheses represents the total available space in the heap.

3. Analyze garbage collector performance.

   The last step is to run configuration management for a few minutes under maximum load and analyze the GC output, watching for the following information:

   - The time taken by full collections. Tuning GC options can reduce pauses that are longer than 5 seconds.
   - The time taken by minor collections. Generally, pauses that are longer than 0.3 seconds can also be reduced.
   - The long-term memory footprint. If full collections always result in a heap size that is less than 20% full, the maximum heap size setting can be lowered. This reduces the pause duration of a full collection.
   - The time between full collections. If full collections occur very frequently, tuning GC options usually helps.

   You can also analyze GC performance by enabling the following flags:

   - `–XX:+PrintGCTimeStamps` to prefix the time of the GC to each entry.
   - `–XX:+PrintGCDetails` to provide additional details about the GCs.
   - `–XX:+PrintTenuringDistribution` to add object aging distribution. This can be very useful when tuning the new generation size and promotion policies.
   - `–XX:+PrintHeapAtGC` to show heap statistics at GC.
   - `-XX:+PrintHeapUsageOverTime` to print heap usage and capacity with timestamps.

Integrity Lifecycle Manager server Caches Too Small

When the server SI or IM metadata cache or bulk data cache is too small for the current usage, overall performance can be reduced. This should be dealt with as soon as possible.

When a server metadata cache is too small, the server log file contains warning messages such as:

```
Warning:
Two successive cache collections in less than 60 minutes
(15 minutes).
If this happens regularly, you may have to allocate more memory
to the cache.
```

In this case, the cache size should be increased by modifying the setting for:

`si.ServerCache.default.size` or `im.ServerCache.default.size`.

When the server is running both configuration management and workflows and documents, make sure you update the proper one. The log file always logs messages when a cache collection begins:

```
SIRootCache Object Collection Begin...
```

Or

```
IMRootCache Object Collection Begin...
```

When the server bulk data cache is too small, the server log file contains warning messages such as:

```
Warning: Two successive bulk-data cache collections in less than
60 minutes (15 minutes).
If this happens regularly, you may have to allocate more disk space
and/or memory to the bulk-data cache.
```

The log file gives information about all cache collections:

```
SIRootCache BulkData Cache Collection Begin...
SIRootCache: Sizes Before:
Memory: current=20Mb (104,15%) high=19Mb(98%) low=16Mb(80%) max=20Mb
Disk: current=2.1Gb(0,11%) high=19Gb(95%) low=16Gb(80%) max=20Gb
```

From this information, you can determine which high-level mark was reached. The first line shows the amount of memory used by the bulk data cache, while the second line shows disk usage. For more information, see the properties for:

- `si.ServerCache.default.bulkMemSize`
- `si.ServerCache.default.bulkDiskSize`
- `im.ServerCache.default.bulkMemSize`
- `im.ServerCache.default.bulkDiskSize`

Integrity Lifecycle Manager server Cache Journal Too Small (Configuration Management Only)

Integrity Lifecycle Manager servers with a local configuration management repository maintain a journal of all events. This journal allows clients and proxies to be brought into sync after a network disruption or after a reboot of the proxy or server. Maintaining a large number of journal entries allows for a longer outage without having to completely clear downstream caches.

If the repository is accessed by a proxy, it is important that the journal is large enough to allow for a long outage. For example, it could take a significant amount of time for a remote site to rebuild a 50-MB, 80-MB, or 120-MB cache. With isolated users on slow connections, ensure that these client caches do not need to be flushed because of a network disruption.

If only LAN users access the Integrity Lifecycle Manager server, there is a reduced impact on performance, and the default settings can be retained.

When the time to renew the journal is less than the proxy's desired outage resistance of 1 or 2 days, the journal size is too small. To allow for a margin of error, you can set a larger journal in the event an outage occurs on a busy day.

You can use log file messages to estimate the turnover frequency of the journal. Find all lines that include:

```
Truncating SI Journal table
```

and then calculate how long the server was running before the journal renewed itself. For example, the following journal renews itself in 2 days:

```
Truncating SI Journal table: first event: 14000, last event: 19000 low
water: 3000, high water: 5000, deleting all events before: 16000
Truncating SI Journal table: first event: 16000, last event: 21000 low
water: 3000, high water: 5000, deleting all events before: 18000
Thu Nov 23 17:00:00 CET 2002:<...>Truncating SI Journal table: first event:
18000, last event: 23000 low water: 3000, high water: 5000, deleting all
events before: 20000
```

Poorly Tuned Executor Threads Can Slow Down the Integrity Lifecycle Manager server

A pool of threads is dedicated to the execution of all incoming requests on a server, whether this is an update to the repository (example: lock, checkin, checkpoint) or whether this is a request for data (file contents or metadata).

This pool is configured to grow as large as necessary up to a default maximum of 300.

It is deliberately set to a high value on the assumption that it only grows as large as actually necessary.

It is still possible that the pool is not sized properly. Look for the following indicators:

- The overall performance of the server decreases, and the CPU is very high. It is possible that the server machine cannot accommodate the load induced by all executor threads.

  This is an indicator that the server may perform better with a smaller executor thread pool.
- Under maximum load, the server CPU activity remains low. A stack dump shows that the pool reached its maximum and all executor threads are always busy.

  This is an indicator that the server may perform better with a larger executor thread pool.

Adjusting Integrity Lifecycle Manager server Settings

This section contains information about adjusting the settings on the Integrity Lifecycle Manager server.

Note

New settings take effect immediately on restarting the server.

Heap Size Configuration	
UNIX, Solaris, Linux, Windows	*server install dir*/config/mksservice.conf *server install dir*\config\mksservice.conf
Configuration Management Settings	
UNIX, Solaris, Linux, Windows	*server install dir*/config/properties/si.properties *server install dir*\config\properties\si.properties
Workflows and Documents Settings	
UNIX, Solaris, Linux, Windows	*server install dir*/config/properties/im.properties *server install dir*\config\properties\im.properties
Common Settings	
Unix, Solaris, Linux, Windows	*server install dir*/config/properties/is.properties *server install dir*\config\properties\is.properties

Adjusting the Database

Various database adjustments may be done to achieve optimal performance. For more information, see Database Performance and Tuning on page 67.

Adjusting Integrity Lifecycle Manager server Heap Size

For the location of configuration files, see Adjusting Integrity Lifecycle Manager server Settings on page 36.

- `-Xmx<max heap size>` (default 1024 Mb)

  Specifies the upper limit of memory assigned to the Java heap.

  Once this limit is reached, no more memory is assigned to the heap, even if the machine has additional memory resources available. For example, a setting of `-Xmx512m` sets the maximum server heap size to 512 MB. Allocating the correct heap size to the Integrity Lifecycle Manager server is critical to good performance.

  Guidelines:

  - Allocate sufficient memory for efficient operations, while not approaching the physical memory limit to avoid disk swapping.
  - Integrity Lifecycle Manager server can only be installed on a 64-bit platform. The theoretical limit for the Xmx value is quite large. Despite this, the Xmx value should be set significantly lower than the available physical memory installed on the server machine. This ensures that there is enough memory available for the operating system as well as other applications running on the same server. The recommended Xmx value is 1024 MB or higher.

    Note

    Setting an Xmx value that is too large may cause the garbage collector (GC) to run longer. If the GC runs longer, the server may become unresponsive for a longer period of time. See the section Java Garbage Collection Overview on page 13.

  The following key facts can help you estimate the current requirements and understand how they may evolve over time:

  - All connected Integrity Lifecycle Manager clients, including web clients, require both heap and native memory.
  - Both the root metadata and bulk data caches use heap memory (as defined by the settings for `size` and `bulkMemSize`).

- The Integrity Lifecycle Manager server requires significant free memory for temporary operations.

- `-Xms<starting heap size>` (default 1024 Mb)

  Specifies the initial amount of memory dedicated to the Java heap when the program starts.

Adjusting Integrity Lifecycle Manager server Garbage Collector Parameters

General notes:

- Tuning GC parameters requires precision and should only be attempted when the GC has proven to be a noticeable factor in slowing performance. For more information, see Garbage Collector Slowing Down the Integrity Lifecycle Manager server on page 31.
- When working with large heap sizes, trade-offs are sometimes required to set the most effective values. It may not always be possible to improve GC times with the current algorithms.
- Testing is the only way to confirm which GC parameters should be used.
- `-Xmx` and `-Xms`

  Maximum and minimum heap sizes (as previously explained).

  The maximum and minimum settings have an impact on GC performance. A larger heap size results in less frequent but longer pauses during garbage collection. If the GC analysis suggests that the heap is too large for your requirements and is slowing the server, then the heap size can be reduced.

> **Note**
>
> Some settings may not be in the configuration files by default. If the value needs to be changed from the default, the new value may need be added to the file manually.

- `-XX:NewRatio` (All operating systems=8)

  The ratio between the OLD and NEW generations (OLD=NewRatio * NEW).

  Guidelines and trade-offs:

  - The most often suggested value for yielding good GC results is 8. (A `NewRatio` of `8` means: NEW=¼ HEAP and OLD=¾ HEAP.) A large Eden generally decreases pause times.

- If you are using a multi-processor server, increase the size of the young generation because allocations are performed in parallel. This is the case even when the GC is not parallel.
- If the new generation is too small, short-lived objects are promoted to the old generation, where they stay until a full GC occurs.
- If the new generation is too large, minor collections result in longer application pauses and the space for long-lived objects can become too small.
- When **ms** equals **mx**, the new generation size is fixed.

- `-XX:MaxNewSize` (default Sparc=32M, Intel=2.5M)

  The maximum size for the new generation.

  The value for `MaxNewSize` limits the growth of the new generation. The default value is often too small and therefore takes precedence over `NewRatio`.

  Note

  The value for `MaxNewSize` should never exceed 50% of the heap size.

- `-XX:NewSize` (default Sparc=2.125M, Intel=640k)

  The initial size for the new generation.

  By setting `NewSize` and `MaxNewSize` to the same value, one can fix the new generation size even when **ms** and **mx** have different values. This increases predictability but prevents the JVM from compensating if the values are not optimized correctly. The `NewSize` setting is also used when the integral multiple allowed by `NewRatio` (that is, ½, ¼…) does not provide sufficient precision.

- `-XX:SurvivorRatio` (All operating systems=8)

  The ratio of each survivor space against Eden (EDEN=SurvivorRatio * S1).

  Generally, the `SurvivorRatio` parameter only has a minor effect on performance.

  A value of `8` is generally recommended for good GC results on the Integrity Lifecycle Manager server (that is, Eden=1/2*NEW, S=1/4*NEW). If the survivor areas are too small, short-lived objects are promoted to the old generation where they stay until a full GC occurs.

- `-Xincgc` and `-Xnoincgc` (default off)

  Enables the train algorithm for OLD generation collections.

An incremental algorithm (train) can replace the default algorithm (mark & compact). The alternate train algorithm collects chunks of memory (one car at a time) instead of the whole heap. This algorithm reduces maximum pause times, but involves a larger GC overhead (by approximately 10%). Using the train algorithm can also lead to a significant performance degradation when memory is low.

- `-XX:+CMSIncrementalMode` and `-XX:+CMSIncrementalPacing`

  Results in the following:

  - Pause during concurrent phases
  - Adjusts duty cycle based on JVM stats
  - Primarily for platforms with 1 CPU to avoid long pauses

  This option may be useful if an unacceptable pause is seen on CMS collection, or on single CPU environments.

- `-XX:+MaxTenuringThreshold` (default=15) and `-XX:+InitialTenuringThreshold` (default=7)

  Number of times an object is aged before promotion to tenured generation. This prevents shorter-living objects from filling up old generation and then fragmenting old generation when collected. It reduces old generational GC time, reduces likelihood of full GC due to full or fragmented old generation.

  Increases the max in increments if GC logs indicate `threshold = max` at full GC.

> **Note**
>
> Set the following to view tenuring log info: `-XX:+PrintTenuringDistribution`.

  This option may be useful when a full GC is executed due to full or fragmented old gen.

- `-XX:+InitiatingOccupancyFraction(def=92%)` and `-XX:+CMSInitiatingOccupancyOnly`

  Old generation occupancy threshold triggers CMS collection.

  This option may be useful when a full GC is executed due to long-running CMS collection that does not clear old generation before it is full.

  The JVM pause prediction algorithm may not have enough information just after server startup to handle a steep climb in memory. This is because many users are starting to use the system. Try reducing this value for the first CMS

collection only (CMSInitiatingOccupancyOnly) to prevent inaccurate predictions about when to start CMS.

Adjusting Integrity Lifecycle Manager server SI Cache Resources

The parameters for the SI server cache are found in `si.properties`. For the location of configuration files, see Adjusting Integrity Lifecycle Manager server Settings on page 36.

The default settings are noted as comments in this file. The prefix for all Integrity Lifecycle Manager server SI cache settings is `si.ServerCache.default`. The prefix `si.ProxyCache.default` is then used to configure the defaults for all proxy caches when FSA is in use. The prefix `si.Cache.default` can also be used to set the default values for all Integrity Lifecycle Manager server and proxy caches.

The following sections provide information on the tuning for each setting.

- `si.ServerCache.default.size` (default=16M; maximum=2048M)

  The maximum amount of memory the SI server metadata cache is allowed to use.

  The SI metadata cache contains data on the structure and history of projects. Therefore, the server cache should be configured to be large enough to hold the required metadata for projects that are used regularly. Otherwise, server performance will be adversely affected whenever an Integrity Lifecycle Manager client is restarted. In some cases, the server cache can even begin to thrash, especially in cases where client caches are also too small.

  The strict minimum value for the server cache is the size required to contain all metadata for frequently used projects. If space is not configured for additional data, frequently used data is collected outside of the cache.

  Tip

  Using the server log, you should be able to view all actively used projects in succession without triggering a server cache collection. To do this, check the server log for a `Cache Cleanup` entry, for example:

  ```
  SI Root Cache: Full Cache Cleanup: collected 1000 items;
  ending size: 1000 duration= 10ms
  ```

  If a collection has occurred, the server cache is too small.

  Key Points:

- The required size of the SI metadata cache depends on many factors. These factors include the number of projects and members, the average number of revisions per archive, the use of variants and labels, and other factors. For example, day-to-day work on a medium size project usually requires between 4 MB and 10 MB for the metadata cache. Extensive use of the same medium size project could require up to 15 MB. Large projects can require 20 MB, 50 MB, or even more for cache resources.
- Variant or build projects that are used regularly increases the cache requirements for a project. The overhead depends on the structure and history of the project. It is likely that regularly using 8 to 10 variants could double the cache resources needed for a given project. If your process requires the extensive use of variants or build projects, this should be taken into account when configuring the server metadata cache.

- `si.ServerCache.default.bulkDiskSize` (default=1000M)

  `si.ServerCache.default.bulkMemSize` (default=30M)

  Respectively, the maximum disk space and memory allocated to the SI bulk data cache.

  Key Points:

  - The SI bulk data cache accelerates the retrieval of contents for frequently accessed revisions.
  - These frequently accessed revisions are stored on disk (`bulkDiskSize`) and indexed in memory (`bulkMemSize`).

  Each entry uses approximately 300 bytes, independent of the bulk data size (but dependent on the directory/file name). In other words, it takes approximately 30 MB of memory to index 100,000 files.

  - A server cache collection is triggered when either the memory or the disk space used by a bulk data cache reaches a high-water mark. Entries (disk + memory) are removed until both memory and disk space have been reduced to the low water mark. To avoid increasing the wrong setting, PTC recommends that you first confirm which limiting factor was attained.
  - The default settings are generally suitable when starting with new projects. Over time, certain settings will likely need to be increased. You can determine when modifications are required by checking for frequent bulk data cache collection warnings. For more information, see Identifying Server Performance and Memory Issues on page 30.

  Guidelines:

  - If disk space is not a problem, then you can set the bulk data cache sizes to a very large value. In this way, only `bulkMemSize` (or the number of

files) acts as a limiting factor. To modify the bulk data cache size, set `bulkDiskSize` to a large number, while always keeping it below the available free disk space. Then, modify the setting for `bulkMemSize` based on the maximum number of files you would like the bulk data cache to hold. This setting is limited by the memory you can afford to allocate to the bulk data cache.

When using this guideline, you usually only need to increase the `bulkMemSize` setting if the bulk data cache becomes too small at any future time.

- The bulk disk size value is the limit for SI cache resources used at a steady state, but it is not a hard limit. If a hard limit has not been set, the bulk disk size can be temporarily exceeded. To set a hard limit, use the `si.ServerCache.default.bulkDiskHardLimit` property.
- When performance is maximized, successively creating a sandbox for all actively used projects (or variant projects) should not trigger a server bulk data cache collection. Check the server log for a bulk data cache cleanup entry such as the following:

  ```
  Bulk-data Cache: Full Cleanup: collected 100 items;
  ending size (disk/mem): 100/10 duration= 10ms
  ```

  If this type of collection has occurred, it means that the bulk data cache is too small.
- Conversely, if the server can operate for weeks without a bulk data cache collection, the bulk data cache is probably too large for your requirements. While an oversized bulk data cache is not a major problem, it does result in the unnecessary use of memory and disk space.
- After creating a sandbox for each project, you can check the bulk data root directory to determine the current average file size. It is important to note that a single large file is enough to change the average file size considerably. When a project ages, the ratio becomes smaller (files get bigger).
- `si.ServerCache.default.journalHighWater` (default 100000)

  `si.ServerCache.default.journalLowWater (default 80000)`

  These settings determine the cache journal size.

  The cache journal settings are only important when a proxy is accessing the repository or when you have isolated remote users. In general, the journal should be large enough to contain several days of operations so that proxy caches never have to be emptied. Large journals result in a

larger database. You can safely increase the size of the cache journal as long as you do not observe any degradation in performance.

> **Tip**
>
> The low-water parameter only determines the safe length for an outage (in number of events). Therefore, PTC recommends that you retain a relatively high number for the low water parameter. PTC suggests that you also configure the high-water parameter a few thousand events over the low water setting.

Examples:

- Low=80000 High=100000

  Disconnect between events 1 and 79999—you can reconnect until 100000. Min=80000

  Disconnect between events 100000 and 179000—you can reconnect until 200000. Min=80000

- Low=200000 High=220000

  Disconnect between events 1 and 199999—you can reconnect until 220000. Min=200000

  Disconnect between events 500000 and 699999—you can reconnect until 700000.

  Min=200000

- `si.ServerCache.default.bulkDiskHardLimit`

  Sets a hard limit for the bulk data cache.

  The bulk data cache does not typically enforce a hard disk limit, so the bulk data cache can temporarily increase past its maximum size setting when new data is added during background cleanup. The recommended approach is to set the maximum size much lower than the maximum disk size. That approach also helps preserve optimum performance at all times. However, if the server is experiencing major peaks leading to “out of disk space” conditions, it is possible to set a hard limit. When a hard limit is set and reached, the bulk data cache pauses until enough space is freed before adding new data.

SI Cache Settings With an Indirect Impact on Server Performance

- `si.ServerCache.default.bulkRootDir` (default=<si_root>/bulk)

A string that specifies the location of the bulk data cache.

You can maximize performance and improve memory usage by:

- Selecting a fast local drive
- Selecting a different drive from the one where archives are stored
- Using a directory close to the root (shorter names decrease memory usage)

For additional information, see Disk Usage for Maximum Performance on page 48.

Adjusting si.Cache.maxServerPoolSize

The property limits the maximum number of memory threads that are available to the Integrity Lifecycle Manager server to use for cache-related operations during source usage. The maximum number of user spaces in the pool can be customized to support different usage patterns.

When the server pool is too small, user requests are unnecessarily put on hold and performance degradation can occur.

Adjusting Integrity Lifecycle Manager server IM Cache Resources

The parameters for the IM server cache are found in `im.properties`. For the location of configuration files, see Adjusting Integrity Lifecycle Manager server Settings on page 36.

The default settings are noted as comments in this file. The prefix for all Integrity Lifecycle Manager server IM cache settings is `im.ServerCache.default`. The prefix `im.ProxyCache.default` is then used to configure the defaults for all IM proxy caches when FSA is in use. The prefix `im.Cache.default` can also be used to set the default values for all Integrity Lifecycle Manager server and proxy caches.

The following sections provide information on the tuning for each setting.

- `im.ServerCache.default.bulkDiskSize` (default=1000M)

  `im.ServerCache.default.bulkMemSize` (default=30M)

  Respectively, the maximum disk space and memory allocated to the IM bulk data cache.

  Key Points:

  - The IM bulk data cache accelerates the retrieval of contents for frequently accessed attachments (both regular and attachments used in rich text fields).

- These attachments are stored on disk (`bulkDiskSize`) and indexed in memory (`bulkMemSize`).

Each entry uses approximately 300 bytes, independent of the bulk data size (but dependent on the directory or file name). In other words, it approximately takes 30 MB of memory to index 100,000 files.

- A server cache collection is triggered when either the memory or the disk space used by a bulk data cache reaches a high-water mark. Entries (disk + memory) are removed until both memory and disk space have been reduced to the low-water mark. To avoid increasing the wrong setting, PTC recommends that you first confirm which limiting factor was attained.
- The default settings are generally suitable when starting with a new IM repository. Over time, certain settings will likely need to be increased. You can determine when modifications are required by checking for frequent bulk data cache collection warnings. For more information, see Identifying Server Performance and Memory Issues on page 30.

Guidelines:

- If disk space is not a problem, then you can set the bulk data cache sizes to a very large value. In this instance, only `bulkMemSize` (or the number of files) acts as a limiting factor. To modify the bulk data cache size, set `bulkDiskSize` to a large number, while always keeping it below the available free disk space. Then, modify the setting for `bulkMemSize` based on the maximum number of files you would like the bulk data cache to hold. This setting is limited by the memory you can afford to allocate to the bulk data cache.

  When using this guideline, you usually only need to increase the `bulkMemSize` setting if the bulk data cache becomes too small at any future time.
- The bulk disk size value is the limit for IM cache resources used at a steady state, but it is not a hard limit. If a hard limit has not been set, the bulk disk size can be temporarily exceeded. To set a hard limit, use the `im.ServerCache.default.bulkDiskHardLimit` property.
- If the server can operate for weeks without a bulk data cache collection, the bulk data cache is probably too large for your requirements. While an oversized bulk data cache is not a major problem, it does result in the unnecessary use of memory and disk space.

- `im.ServerCache.default.bulkDiskHardLimit`

  Sets a hard limit for the bulk data cache.

The bulk data cache does not typically enforce a hard disk limit, so the bulk data cache can temporarily increase past its maximum size setting when new data is added during background cleanup. The recommended approach is to set the maximum size much lower than the maximum disk size. That approach also helps preserve optimum performance at all times. However, if the server is experiencing major peaks leading to “out of disk space” conditions, it is possible to set a hard limit. When a hard limit is set and reached, the bulk data cache pauses until enough space is freed before adding new data.

- `im.ServerCache.default.size` (default=16M)

  The maximum amount of memory the server metadata cache is allowed to use.

  At the current time, the IM metadata cache is only used as a helper to the IM bulk data cache. It contains few data, like the identity of attachments for ACL checks. Therefore, there is rarely the need to tune it. If you see regular warnings from the IM server root cache, adjust this setting.

IM Cache Settings With an Indirect Impact on Server Performance

- `im.ServerCache.default.bulkRootDir` (default=<im_root>/IMBulk)

  A string that specifies the location of the bulk data cache.

  You can maximize performance and improve memory usage by:

  - Selecting a fast local drive
  - Using a directory close to the root (shorter names decrease memory usage)

  For additional information, see Disk Usage for Maximum Performance on page 48.

- `mksis.rmi.maxExecutorThreads` (default=300)

  The maximum number of threads in the executor pool can be customized. The setting limits the maximum number of simultaneous operations the server is able to fulfill.

Adjusting the Integrity Lifecycle Manager server Cache Notification Setting

When notifying connected clients about cache changes, the Integrity Lifecycle Manager server creates a new thread pool and notifies Integrity Lifecycle Manager clients using that pool for each cache change notification.

The `mksis.im.cacheNotificationPoolSize` property controls the maximum number of threads available to the Integrity Lifecycle Manager server for notifying Integrity Lifecycle Manager clients about cache updates. Cache update notifications ensure that clients update their caches to reflect current data on the Integrity Lifecycle Manager server. If users report that they are not seeing current data on the Integrity Lifecycle Manager server, their clients may not have received cache update notifications yet.

Consider increasing the number of threads to improve the Integrity Lifecycle Manager server's ability to handle the rate at which cache notifications are generated and sent to connected clients. Depending on the selected value, this setting has the potential to affect server performance and result in duplicate notifications. Before you increase the number of threads, PTC recommends that you analyze the server statistics for the IM admin cache notification pool. This analysis can help determine if the value needs to be increased.

By default, `mksis.im.cacheNotificationPoolSize=50`.

Guidelines:

The number of threads should be increased when some, or all, of the following conditions are occurring on a daily basis:

- The average `active thread count` is large and close to the defined maximum number of threads.
- The average or maximum `backing queue size` is high.
- The average notification processing time is high, given the number of users connected to the system.

  These conditions can arise when numerous Integrity Lifecycle Manager administrative changes are made to the system over a short period of time. This also is the case when working extensively with item backed pick lists or if the number of users connected to the server is significant.

  To monitor conditions on the Integrity Lifecycle Manager server, you can use the `im stats --target=server` command and refer to the “Stats” section of Performance Analysis Tools on page 26.

For general information on configuring Integrity Lifecycle Manager server properties, see the *Integrity Lifecycle Manager Installation and Upgrading Guide*.

Disk Usage for Maximum Performance

Key Points:

- Use large drives—disk space is not expensive. Ensure that there is always plenty of free disk space available or the application may be slowed down.
- Use a fast drive—a fast drive improves archive and bulk data cache accessing. Avoid using networked drives whenever possible.
- Use several disks—on the server, place the repository and the bulk data cache on different drives to reduce the load on each disk. Similarly, on the proxy, the proxy bulk data caches can be distributed across different drives.

5

Configuring and Tuning the Proxy

This section applies when tuning a proxy (that is, when FSA is enabled). The proxy is an Integrity Lifecycle Manager server. Therefore, many guidelines outlined in the previous section apply equally to the proxy. This section outlines the attributes that are unique to the proxy.

Identifying Proxy Performance and Memory Issues

This section provides information and instructions for identifying issues with the performance of the proxy server. This section also discusses identifying memory issues.

Proxy Memory and GC Issues

The procedures for detecting memory issues in the proxy are the same as for the Integrity Lifecycle Manager server. For information on these topics, see Identifying Server Performance and Memory Issues on page 30 or Garbage Collector Slowing Down the Integrity Lifecycle Manager server on page 31.

Proxy Cache Too Small

Check for warning messages. If you have multiple proxies, you need to check which specific proxy caches are too small.

SI ProxyCache_Chicago:

Full Cache Cleanup: collected 1000 items; ending size: 1000…

Warning: Two successive cache collections in less than 60 minutes (15 minutes). If this happens regularly, you may have to allocate more memory to the cache.

In this situation, the corresponding cache (or bulk data cache) size should be increased.

Poorly Tuned Executor Threads Can Slow Down the Proxy

The proxy acts as a server for incoming client requests and as a client for each outgoing server operation. Proxies use more `executor threads` than standard servers, therefore it is important that there are enough threads. Carefully observing for signs described in Configuring and Tuning the Integrity Lifecycle Manager server on page 29 confirms this.

Adjusting Proxy Settings

This section provides information on adjusting the heap size, GC parameters, and other settings for a proxy.

Adjusting the Proxy Heap Size

The locations of the files are the same as with any Integrity Lifecycle Manager server; however, the memory requirements can be quite different. For example, a single server can proxy several remote servers. Furthermore, when used in a global environment, sites are often connected by a slow network. As a result, resources allocated to caches are typically higher to avoid (as much as possible) transferring the same data twice.

To help you estimate current requirements and provide suggestions about their likely evolution over time, always consider the following:

- The requirements of all components are compounded (local server + all proxies).
- A single user accessing several repositories through a proxy (including the local repository) counts multiple times.
- Each set of proxy caches uses heap memory for both metadata and bulk data. This consists of the sum of all `size` and `bulkMemSize` settings for all proxy components, as well as the local component.
- Just as for a standard Integrity Lifecycle Manager server, the proxy requires sufficient free memory for temporary operations.

Adjusting Proxy Garbage Collection Parameters

Adjusting the GC parameters for the proxy follows the same procedures as for the standard Integrity Lifecycle Manager server.

For more information, see Adjusting Integrity Lifecycle Manager server Garbage Collector Parameters on page 38.

Adjusting Proxy Cache Resources

This section focuses on proxy cache settings. If the proxy also has local projects, you can configure the cache by following the procedures outlined in the previous section entitled Adjusting Integrity Lifecycle Manager server Settings on page 36.

The prefixes for proxy cache settings are:

- `si.ProxyCache.default.` to assign a default value to any SI proxy cache.
- `si.ProxyCache.<alias>.` to assign a value for a particular proxied SI server.

- `im.ProxyCache.default.` to assign a default value to any IM proxy cache.
- `im.ProxyCache.<alias>.` to assign a value for a particular proxied IM server.

(alias is the name of the proxy as it appears in `is.properties: mksis.proxyList`)

- `si.ProxyCache.<alias>.size (default=16M)`

  `si.ProxyCache.<alias>.bulkMemSize (default=30M)`

  `si.ProxyCache.<alias>.bulkDiskSize (default=1000M)`

  `si.ProxyCache.<alias>.bulkDiskHardLimit (default=0, where 0 is no limit)`
- `im.ProxyCache.<alias>.size (default=16M)`

  `im.ProxyCache.<alias>.bulkMemSize (default=30M)`

  `im.ProxyCache.<alias>.bulkDiskSize (default=1000M)`

  `im.ProxyCache.<alias>.bulkDiskHardLimit (default=0, where 0 is no limit)`

As indicated previously, it is important to have a cache that is large enough to hold frequently used data. This principle is especially true for the SI proxy bulk data cache. The cache settings can make the difference between a resync on a given sandbox that takes three minutes and one that takes an hour.

Other Settings With an Impact on Proxy Performance

- `si.ProxyCache.<alias>.rootDir (default=<si_root>/<alias>/Metadata)`

  `si.ProxyCache.<alias>.bulkRootDir (default=<si_root>/<alias>/Bulk)`
- `im.ProxyCache.<alias>.rootDir (default=<im_root>/<alias>/IMMetadata)`

  `im.ProxyCache.<alias>.bulkRootDir (default=<im_root>/<alias>/IMBulk)`

Respectively, the location of the metadata and bulk data proxy caches.

`RootDir` specifies the location where the proxy metadata cache is saved regularly.

You can maximize performance and improve memory usage by:

- Selecting a different drive for each remote server and for the root bulk data cache, if applicable
- Selecting fast local drives
- Using directories that are located close to the root since shorter names decrease memory usage

Adjusting Other Integrity Lifecycle Manager server Settings

General settings for the Integrity Lifecycle Manager server are found in `is.properties`. For the location of configuration files, see Adjusting Integrity Lifecycle Manager server Settings on page 36.

- `mksis.default.compressionEnabled (default=true)`

  `mksis.<alias>.compressionEnabled (default=true)`

  These settings specify whether the proxy should use compression when communicating with end servers. The `mksis.default.compressionEnabled` setting enables compression for all proxies, while the `mksis.<alias>.compressionEnabled` setting enables compression for a single proxy.

  In general, because most proxy/server connecting networks are suboptimal, compression should be enabled. When using a proxy for load balancing on a LAN, compression may actually slow down communications and therefore can be disabled.

- `mksis.rmi.maxExecutorThreads(default=300)`

  The maximum number of threads in the executor pool can be customized. The setting limits the maximum number of simultaneous operations the server is able to fulfill.

Adjusting Other Integrity Lifecycle Manager client Settings

These Integrity Lifecycle Manager client settings can be set in `IntegrityClientSite.rc`. For the location of the file, see Adjusting Client Settings on page 63.

- `mks.default.compressionEnabled(default=false)`

  `IntegrityClient.`*`hostname`*`.`*`port`*`.compressionEnabled (default=false)`

These settings specify whether the client should use compression for communications between the Integrity Lifecycle Manager client and the closest server (proxy or end server for a direct connection).

The `mks.default.compressionEnabled` setting enables compression for all connections, while the `IntegrityClient.`*`hostname`*`.`*`port`*`.compressionEnabled` setting enables compression for a single connection.

SI Priming for a Specified Project

As administrator, you can use the `si primeproject` command to manually prime the proxy for a new project before users access that project through the proxy. For larger projects, priming the proxy in advance means that users are not delayed by waiting for that project's data to download from the host.

When you run the `si primeproject` command, the proxy downloads bulk data for the project you have specified. Once you have primed the proxy for a specified project, performance is improved for users when subsequently creating the associated normal, variant, or build sandbox.

You can also use the `si primeproject` command any time there is significant new data for an individual project on the host. Priming for a specified project can be carried out regularly to maximize performance for users connecting to a proxy. For more information on the `si primeproject` command, see the CLI man pages.

Automatic Population of the SI Bulk Data Cache

For known projects, the proxy automatically downloads bulk data for newly checked-in revisions or added members. When a user does a checkin or adds a new member, the bulk data cache fetches the corresponding revision contents, anticipating the next resync or checkout. The proxy carries out automatic downloads by default, but this behavior is configurable by modifying the following property in the `si.properties` file on the proxy machine:

```
si.default.autoFetchNewRevisions
```

If set to true (the default setting), this property allows the proxy to automatically download bulk data for newly checked-in revisions or added members. If set to false, the proxy does not automatically download bulk data for new revisions. As administrator, you can modify the behavior of the proxy by configuring the setting for true or false.

This feature does not carry any network traffic limitation. On a system with many updates but with a very low bandwidth, it may be preferable to set the value to false. This can be applied globally or for a specific proxy. For more information on setting the `si.default.autoFetchNewRevisions` property, see the *Integrity Lifecycle Manager Installation and Upgrading Guide*.

6

Configuring and Tuning the Integrity Lifecycle Manager client

This section provides information and instruction for identifying problems with the Integrity Lifecycle Manager client as well as adjusting the settings for optimal performance. These adjustments include heap size, cache resources, and database administrative tasks.

Identifying Client Performance and Memory Issues

This section provides guidance for identifying signs of suboptimal tuning in the Integrity Lifecycle Manager client through log files analysis and client behavior checks.

Location of Client Log Files

The client log file is called `IntegrityClient.log`:

- UNIX (Solaris, Linux)

  client install dir`/bin/IntegrityClient.log`

- Windows

  client install dir`\bin\IntegrityClient.log`

> **Tip**
>
> The Integrity Lifecycle Manager client log file is emptied after startup. If you observe an error message, check the log file before you restart the Integrity Lifecycle Manager client. You may want to create a backup copy of this log file for future reference.

Integrity Lifecycle Manager client Out of Java Heap Memory

When the Integrity Lifecycle Manager client is close to running out of heap memory, the Java garbage collector (GC) runs frequently causing the application to slow down. It may also show temporary freeze conditions. When the Integrity Lifecycle Manager client is actually out of heap memory, pop-up error windows are displayed with messages such as shown in the following example.

```
Not enough memory: The Integrity Client is out of memory.
Contact the Integrity Administrator for the corrective action.
java.lang.OutOfMemoryError
<<no stack trace available>>
```

The log file contains error entries such as:

```
<Adapter> OutOfMemoryError in Adapter
java.lang.OutOfMemoryError
<<no stack trace available>>
```

Or:

```
Not enough memory: The Integrity Client is out of memory.
Contact the Integrity Administrator for the corrective action.
```

```
at mks.frame.cache.Cache.<clinit>(Cache.java:50)
at mks.si.pinky.api.PropertiesCache.createCache(PropertiesCache.java:240)
at mks.si.pinky.api.PropertiesCache.init(PropertiesCache.java:199)
```

If you see these types of messages regularly, the requirements of the Java heap are greater than the current settings. To address this problem, increase the maximum heap size for the Integrity Lifecycle Manager client.

If you only see such messages infrequently—for example, when working with a particularly large amount of data—there is likely no need for any changes.

Key Points:

- A common mistake is to assume the second error message means that cache resources are too small (because the stack trace contains a reference to cache). Out of memory means out of heap, regardless of the circumstances. When the cache is too small, there are messages indicating this. For more information, see the section entitled Integrity Lifecycle Manager client SI Cache Too Small on page 62. In this situation, allocating more resources to the cache would only make the problem worse, because the client would use even more memory.
- Given the distributed architecture of configuration management, a lack of memory in the server is sometimes propagated to the Integrity Lifecycle Manager client. Administrators may be notified of this problem, but there is no requirement to adjust client settings in this case.

Integrity Lifecycle Manager client Out of Physical or Virtual Memory

When available physical memory runs out on the client computer, all programs slow down and the operating system uses the hard drive for virtual memory. When the virtual memory itself is exhausted, warning messages from the operating system are displayed. In addition, some applications freeze or shut down.

If there is a problem with physical or virtual memory, confirm whether there are too many applications running. If there are not too many applications running, it is likely that the client computer requires additional memory to support the current Integrity Lifecycle Manager client requirements.

Alternatively, the Integrity Lifecycle Manager client may have too much memory allocated to the heap compared to the available virtual memory. To address this problem, check the Integrity Lifecycle Manager client heap settings.

Integrity Lifecycle Manager client SI Cache Too Small

When the client cache is too small for current usage, performance degrades and data must be reacquired from the server cache at regular intervals.

Check the client log files for warning messages such as the following:

Chicago_IS:

Full Cache Cleanup: collected 1000 items; ending size: 1000…

Warning:

Two successive cache collections in less than 60 minutes (15 minutes). If this happens regularly, you may have to allocate more memory to the cache.

Configuration management includes a mechanism that prevents thrashing in the client cache when opening new GUI views and when using the command line interface or scripts. This mechanism prevents data dropping from the client cache when it is too small to receive the requested information. The mechanism helps to reduce network traffic and the load on the server/proxy. The cache governor automatically increases maximum size of a cache to fit the data required for all opened views. When the Integrity Lifecycle Manager client is restarted, the cache is automatically restored to its original size. This feature allows users to occasionally work with more views than they usually do, without having to update their settings or experience poor performance. Although the cache is designed to increase to accommodate occasional peaks, it is still necessary to adjust the individual Integrity Lifecycle Manager client settings for more demanding requirements. You should increase the size of the corresponding client cache if you regularly see messages such as the following:

Cache too small. Current Size=1499500, Current Max Size=15000000, New Max Size=16000000

Tokyo_Dev:

Cleanup Failure: collected 1000 items; ending size: 16000000, cache grown

When working against multiple servers, it is also important to verify that all of the client caches are large enough to meet your requirements. Do not immediately modify the setting for a well-tuned client cache. Instead, you should carefully check the log file for the *alias* name to confirm which specific client cache is causing the problem. For example, in the preceding sample logs, there are problems with the caches for `Chicago_IS` (two cleanups in less than 60 minutes) and `Tokyo_Dev` (cleanup failure).

To increase the client cache, adjust the property for `si.ClientCache.`*`alias`*`.size`. For more information, see Adjusting Client Settings on page 63.

Integrity Lifecycle Manager client IM Bulk Data Cache Too Small

When the IM client bulk data cache is too small for current usage, performance degrades when viewing and editing items with rich fields.

If performance degrades, check the client log files for warning messages such as the following:

Warning:

Two successive bulk-data cache collections in less than 60 minutes (15 minutes).

If this happens regularly, you may have to allocate more disk space and/or memory to the bulk data cache.

The log file gives information about cache collections:

IMUserCache_hostname^port^user BulkData Cache Collection Begin...

IMUserCache_hostname^port^user: Sizes Before:

Memory: current=20Mb (104,15%) high=19Mb(98%) low=16Mb(80%) max=20Mb

Disk: current=2.1Gb(0,11%) high=19Gb(95%) low=16Gb(80%) max=20Gb

From this information, you can determine which high-level mark was reached. The first line shows the amount of memory used by the bulk data cache, while the second line shows disk usage.

Adjusting Client Settings

This section covers adjusting client settings. In addition to adjusting heap size and SI cache resources, this chapter provides information on

Configuration File Locations

Heap Size Configuration	
UNIX, Solaris, Linux, Windows	*client install dir*/bin/IntegrityClient.lax *client install dir*\bin\IntegrityClient.lax
Section for Additional Options	
#LAX.NL.JAVA.OPTION.ADDITIONAL	lax.nl.java.option.additional=...

All Other Settings	
UNIX, Solaris, Linux, Windows	*client install dir*/IntegrityClientSite.rc *client install dir*\IntegrityClientSite.rc

Adjusting the Client Heap Size

- `-Xmx<maxheap size>` (default 512 Mb)

  Specifies the upper limit of memory assigned to the client Java heap.

  Once the specified limit is reached, no more memory is assigned to the heap, even if there are additional memory resources available on the client computer.

  For example, **-Xmx92m** sets the maximum client heap size to 92 MB.

  Allocating the correct client heap size is very important. As a general guideline, you must allocate sufficient memory for smooth operations, while staying away from the physical memory limit to avoid disk swapping.

  Note

  After changing the Xmx value, the client must be restarted for the settings to take effect. If you have increased the Xmx value, and the client does not start, then the value is too large. Try decreasing the Xmx value. It is recommended that the value not be lower than 512 MB. For better performance, and if the computer resources can accommodate it, the recommended Xmx value is 1024 MB.

  The following information can help you to estimate the client's requirements and understand how those requirements may evolve over time:

  - Large GUI views demand a large amount of memory.
  - Cache and user requirements increase with the number of connections. For example, when accessing two end servers, requirements are doubled. Similarly, when accessing two end servers through a single proxy, requirements are doubled. Also, when using two logins to connect to the server (for example, *user* and *adminUser*), requirements are doubled.

- `-Xms<starting heap size>`

  Specifies the initial amount of memory dedicated to the Java heap when the program starts.

As the program runs, the heap grows, as needed, up to the maximum size. Setting a low initial heap size prevents the program from allocating memory that it might not otherwise need.

Alternatively, you could set the initial size to the maximum size. This increases predictability and reduces any slowdowns resulting from successive growth of the heap.

Adjusting SI Client Cache Resources

The prefixes for client cache settings are:

- `si.ClientCache.default.` to give a default value to any client cache.
- `si.clientCache.`*`serverHost`*`.`*`serverPort`*`.` to override a specific end server.
- `si.ClientCache.`*`serverID`*`.size` (default=8M)

  The maximum amount of memory allocated to a client metadata cache.

  The metadata cache contains data on the structure and history of projects. Always try to maintain a cache size that is large enough to contain everyday data. If this is not done, performance is reduced when switching from one view to another. Also note that this is a strict minimum—if you do not allow for additional data, frequently used data is collected outside the cache.

> **Tip**
>
> A typical user session should proceed without triggering a cache collection and without the cache growing. If, during a typical session, cache collections are triggered or the cache grows, one of the client caches is too small for the current requirements.

The size of the metadata cache can be difficult to estimate because it depends on many factors. These factors include the number and size of projects users are working with, the use of graphical views, and other factors. For example:

- Day to day work on a medium-sized project usually requires between 4 MB and 10 MB of cache. Extensive use on the same project could require 15 MB.
- Large projects can require 20 MB, 50 MB, or even more, for memory resources. For larger projects, PTC recommends that users create their sandbox based on subprojects.

Caution

When client cache resources are not sufficient, this also has the side effect of increasing stresses on the server cache. Thus, several poorly tuned clients can place additional demands on the server.

IM Cache Settings With an Indirect Impact on Client Performance

`im.ClientCache.default.bulkRootDir=<user.home>/.mks/ci/bulk`

The previous string specifies the location of the bulk data cache.

You can maximize performance and improve memory usage by:

- Selecting a fast local drive
- Using a directory close to the root (shorter names decrease memory usage)

For additional information, see Disk Usage for Maximum Performance on page 48.

Other Settings With an Impact on Performance

`IntegrityClient.default.compressionEnabled` (default=false)

`IntegrityClient.`*`serverID`*`.compressionEnabled` (default=false)

These settings specify whether the client should use compression when communicating with servers and proxies.

In general, it is preferable to enable compression whenever the Integrity Lifecycle Manager client is communicating with a server or proxy through a slow network connection.

7

Database Performance and Tuning

This section presents options for improving database performance. It addresses administrative tasks and suboptimal query performance as well as triggers and issues tables.

Database Administrative Tasks

Integrity Lifecycle Manager servers use the database heavily. As such, and like any database intensive application, there is the need for a minimum of database administration. This administration helps preserve good performance as the size of the repository increases. This is true for servers configured for workflows and documents as well as configuration management servers. This administration is imperative when running the database back end for configuration management servers.

- Use statistics

  Your database needs to be analyzed periodically.

  All databases maintain statistics about the data in various tables. If the statistics are not up to date, then the plans generated by the databases are too slow, thus making queries slower.

  This is very obvious during an initial database load, if you are migrating data into the system. During a migration, if you analyze the database the migration proceeds much faster. Most databases do update their statistics nightly. Check with your administrator to verify that these updates are happening. In Oracle, the decisions made by the default nightly run on what statistics to compute are not adequate. An occasional full forced statistics computation is required. For information on how to update statistics on an Oracle database, see the Knowledge Base section of the Integrity Lifecycle Manager Help Center.

  You can also cause the statistics to be run via the diagnostic:
  `im/si diag --diag=computestatistics`

  Caution

  The `computestatistics` command can seriously affect system performance, especially if you have a large database. It should be run at a time when the decreased system performance will not affect users. As an alternative, you can run this command on a specific table instead of globally by using the `--param <tablename>` option.

- Use multiple table spaces

  The following is only applicable to Oracle. Some of the tables in the Integrity Lifecycle Manager database store large objects (LOBs). Storing LOBs in different table spaces may improve performance. In particular, consider the following tables: `CMARCHBULK(value)` and `Attachments`.

If you are unsure how to take any of the preceding actions, review the documentation provided by the database vendor you are using. You can also work with your database administrator.

Performance of Integrity Lifecycle Manager Queries

This section deals with slow running IM queries, one of the most frequent performance issues in Integrity Lifecycle Manager (IM).

Detecting Suboptimal Queries

To determine which IM queries are slow, run `im stats` and look for the [Per Query] section.

[Per Query]						
Defects to be Verified	7	22	3	4	2	(#issues)
Defects to be Verified	7	5448463	778351	1729368	56158	(Ms)

Each query should show up twice here, once with the number of issues returned and once with the amount of time taken, in microseconds.

Queries that have run many times but take more than a few hundred milliseconds are candidates for optimization. In the preceding table, the “Defects to be Verified” query is a candidate for optimization.

Optimizing Individual Queries Running Slowly

- Qualify queries further.

  When an individual query is running slowly, it is frequently because it is not sufficiently qualified and cannot use an index. Most of the time, if you qualify it further, the query can use an index. This reduces the number of non-indexed elements it has to process. (For example: adding “State = Submit” to the query).

  When further qualifying a query does not help, several actions can be taken to address the performance issue. It is the responsibility of the IM administrator working with the database administrator to determine an appropriate manner to solve the problem.

- Create indexes.

  Indexes can be created for the following:

- Fields
- Tables

User-created index names that are the same as the ones created during an upgrade create conflicts and may cause upgrade failure. To avoid conflicts, you can prefix `CUST_` to your index names. For example,`CUST_<TableName>_<columnName1>_<columnName2>...`

- Creating indexes for fields

A possible solution is to add an index on the field. This is only done when the field is frequently used by queries, because too many indexes can slow down the system. To add an index to a field, use the `createissuesindex` diagnostic:

```
im diag --diag=createissuesindex indexName fieldname1 fieldname2 ...
```

For example:

```
im diag --diag=createissuesindex IssuesDueDate 'Due Date'
```

The preceding action can be taken to add an index on the `Due Date` field.

You can also use the normal database tools to create the index.

- Creating indexes for tables

You can add an index to a table. To add an index to a table, use the `createindex` diagnostic:

```
im diag --diag=createindex <index_name> <table_name>
<column_name1> [column_name2 ...] [initial=<initial
extent size in kK/mM/gG>] [next=<next extent size in kK/
mM/gG>]
```

For example:

```
im diag --diag=createindex CUST_FIELD1_SET_ID FIELD1_
SET ID initial=1M next=10M
```

This creates an index of name `CUST_FIELD1_SET_ID` of table name `FIELD1_SET` with column name `ID`, initial extent size as 1 MB, next extent size as 10 MB. Adding of initial and next extent sizes can be only possible in Oracle Database using this diagnostic.

Specifying initial or next extent values while creating an index for table is optional.

Note

An error message is displayed when:

- Invalid values that do not satisfy the syntax for `createindex` diagnostic.
- Any value exists after the parameters, initial or the next extent size of the index.
- Initial and the next extent values are defined on a Database other than Oracle.

- Use static fields.

  Computations that involve the two functions Aggregate and Query, or query-backed relationships are frequently very slow and not easily subject to indexes. They are generally unusable when they appear as part of a query. They can also be problematic when simply being retrieved as a field.

  Static fields are computed on a schedule. In general, the duration is not a problem. Once they have been calculated, they are just columns in the issues table, and hence, they can be indexed.

  Also see “Best Practices and Key Considerations for Computed Expressions” in the *Integrity Lifecycle Manager Installation and Upgrading Guide*.
- Move text fields to user-defined tables.

  Multiple text fields may share a text index. When writing a query, the database needs to determine the best way to access the data. A query that filters against a particular text field may come up with an access plan that requires a lookup in the text index. This query returns all internal items that match. This includes ones in other text fields that have the same text table. If the text search is not specific, then this may slow down your overall query. No additional items are returned. It is all internal to the database. It is hard to diagnose such a problem.

  If this is a problem, you can create a user-defined text table for the fields you need to query on using the following procedure.

Note

- Using user-defined tables for text fields can slow down all-fields text searching (searches all items).
- When creating a new text table, a new lexer can also be created at the same time.

1. Create the user-defined text table.

   Run `im diag --diag=createuserdefinedtexttable`

   For example:
   ```
   im diag --diag=createuserdefinedtexttable 5 clob
   ```
   creates table Text5, with a default text index on the Value column.

   Note the following:

   - You may create a text table with any number from 5-100.
   - The final keyword to the diagnostic creates the table with the Value column as a character large object (clob) or as a variable character field (varchar). Variable character fields are generally more efficient. If all fields using this table have a known, small size, you can use varchar. The maximum size for varchar is database-specific, but generally you can use it for fields less than 4 KB.
   - Multiple text fields can share a table. You may not need to perform this step if you are moving to a user-defined table that you have already created for other purposes.
2. Move the data from the default location.

   Run `im diag --diag=setuserdefinedtexttable`

   For example:

   ```
   im diag --diag=setuserdefinedtexttable fieldname 5
   ```
   moves the data from the default text table to the Text5 table.

Note

If you have a large amount of data in the field to be moved, consider dropping the text indexes on both the source and target tables. After dropping the text indexes, move the data to the user-defined table. You can then re-create the indexes later. Inserting or deleting into or from an indexed table takes much longer than inserting or deleting into an unindexed table. This is true even taking into account the time to recreate the indexes later. This is dependent on the underlying database.

3. Re-create the index.

   If you dropped the index in step 2, re-create it now.

Filtering Queries in the Web Interface

When running a query in the Integrity Lifecycle Manager Web interface, the maximum number of records returned depends on the value set for the `mksis.im.itemTextFilterThreshold` property. This property controls the maximum number of items that can be returned before the **Show items containing** text filter is disabled.

By default, the number of records returned by the text search filter is 1000. Administrators can increase this property value to a maximum of 25000 records in the Integrity Lifecycle Manager administration client by selecting **Workflows and Documents ▸ Configuration ▸ Properties**. The change is applied immediately on the Integrity Lifecycle Manager server. However, current users may need to log out and log in to the Web interface to see the effect of the modified value.

Note

If you choose to increase the `mksis.im.itemTextFilterThreshold` property beyond its default value, PTC recommends that you increase both the `mksis.im.queryGovenor` property and the `mksis.im.queryMemoryGovenor` property to avoid performance degradation.

Setting the Query Timeout

The query timeout is the time after which a query aborts if not completed.

- The default timeout limit is 15 seconds.

It can be changed by using the following diag:

```
im diag --diag=setintpolicy --param=QueryTimeOut --param=<InSeconds>
```

The query timeout can also be set through the Administration GUI by setting the `mksis.im.QueryTimeOut` property under **Workflows and Documents ▸ Configuration ▸ Properties**.

- A different query timeout can be set for each group.

  It can be set in the administration panel for groups or using the following command:

  ```
  im editgroup --queryTimeout=<timeOutInSeconds> GroupName
  ```

Note

- A value of 0 means no timeout is in effect so queries always run to completion.
- When a user belongs to multiple groups, the lower setting is used.

Detecting and Stopping Unusually Long Running Queries

The following diagnostics are a last resort—you should set reasonable query timeouts. Normally a query timeout occurs long before an administrator can find and stop a query.

```
im diag --diag=running
```

This diagnostic can also be run through the Administration GUI under **Workflows and Documents ▸ Diagnostics ▸ Running**.

This `diag` command allows an administrator to detect long running queries.

It lists all running queries together with an identifier in case one of them needs to be stopped using the following `diag`.

Sample output for a running query:

ID:	122787
User:	username
Duration:	300000ms
Status:	RUNNING
Type:	Query
Name:	QueryName
SQL:	SELECT …

```
im diag --diag=kill --param=<threadID>
```

This `diag` command allows an administrator to stop a long running query.

In the preceding example, that would be:

```
im diag --diag=kill --param=122787
```

Performance of Triggers

Sometimes the triggers you have in place contribute to the performance of your operations. This applies to both configuration management and workflows and documents.

The statistics output shows you in the [Triggers] section the time spent in the pre and post invocations of each trigger. It can also show the time spent in the pre and post invocations for a scheduled trigger.

[Triggers]						
Issue changed: pre	744	1143338	1536	47167	794	(Ms)
Issue changed: post	740	279094	377	34536	207	(Ms)
Scheduled: Sample - Hourly	2	133620133	66810066	87903558	45716575	(Ms)

For Integrity Lifecycle Manager, CPU processing time is generally not the problem with a trigger, but the time required can increase with the fetching of other related issues. For example, required time increases with the `imServerBean.getIssueBean` method.

As an alternative, consider using the `imServerBean.getIssueBeans` method that can retrieve multiple issues at the same time, restricted to a subset of fields. When using this method, fetch only the subset of fields your trigger actually requires. Performance may be proportional to the number of fields fetched, and may be significantly slower if you fetch certain computed fields.

Other Performance Improvements

Reducing the Number of Issues Tables

If you have existing data that requires multiple issues tables, you can move columns between issues tables to reduce the number of tables required. You can move the columns required for an index into the same table.

You must update the value of the TableNumber column for the fields you move to the fields table.

Note

Ensure that a database backup is performed and completed before proceeding with this operation. PTC does not provide further assistance in implementing this improvement.

Creating Indexes Across Multiple Issues Tables

If you are using an Oracle database, create an Oracle table cluster and move the issues tables into the cluster. This enables you to create indexes across multiple issues tables.

An Oracle table cluster is a group of tables that share common columns and store related data in the same blocks.

Note

PTC does not help with implementing this improvement. Your database administrator should assess and implement this recommendation.

Changing Values on a Text Table

The default table that is used by a given text field is defined by three values:

- Long text or short text
- Rich content flag
- Text index flag

Note

You should make your decision on these table values before adding data to the field.

If you edit any of these values, the system moves the text data to a new text table. If either table has a text index, the insert and delete operations update the text index on one or both tables.

For better performance, consider dropping the text indexes both on the source table and the target table before performing this operation and re-creating the indexes later. Inserting or deleting into or from an indexed table takes much longer

than inserting or deleting into an unindexed table. This is true even when you take into account the time to re-create the indexes later. This is dependent on the underlying database.

8

JVM Tuning and Oracle SQL*Net Configuration

This section gives information about tuning and configuration for JVM and Oracle SQL*Net. Browse this section for advice on encryption and data integrity verification.

JVM Tuning

Integrity Lifecycle Manager uses Oracle JVM, for which numerous parameters can be tuned. Consult Oracle's JVM documentation for more information.

Oracle SQL*Net Configuration

When you enable encryption or data integrity (checksum) verification on the Integrity Lifecycle Manager server, additional overhead occurs for all communications between the Integrity Lifecycle Manager server and the Oracle database. The amount of overhead depends on the amount of data being sent and on the encryption and data integrity (checksum) algorithms that you choose.

9

Appendices

This section includes helpful information for identifying and solving problems with configuring and tuning. Review this chapter to find quick references, links, and additional help.

Troubleshooting

Symptoms

Q: What action should I take if I regularly receive out-of-memory messages?

This is a serious error message that should be dealt with immediately, particularly if it originates from a server or proxy. For more information, see Identifying Server Performance and Memory Issues on page 30.

Q: The log file contains numerous cache warning messages. What does this mean?

At least one of the caches (whether server, proxy, or client) is too small. For more information, review the following sections:

- For the Integrity Lifecycle Manager server, see Configuring and Tuning the Integrity Lifecycle Manager server on page 29.
- For the proxy, see Configuring and Tuning the Proxy on page 51.
- For the Integrity Lifecycle Manager client, see Configuring and Tuning the Integrity Lifecycle Manager client on page 59.

Q: What action should I take if configuration management is regularly freezing for a few seconds for no obvious reason?

It is possible that one of the Garbage Collectors (GC) is performing poorly. For more information on addressing this problem, see Java Garbage Collection Overview on page 13 and Configuring and Tuning the Integrity Lifecycle Manager server on page 29.

Q: Configuration management sometimes becomes very slow and activity on the hard drive increases at the same time. How can this problem be resolved?

It is possible that the system is swapping for lack of physical memory. For more information, review the relevant sections for the Integrity Lifecycle Manager server, proxy, or Integrity Lifecycle Manager client:

- For the Integrity Lifecycle Manager server, see Identifying Server Performance and Memory Issues on page 30.
- For the proxy, see Identifying Proxy Performance and Memory Issues on page 52.
- For the Integrity Lifecycle Manager client, see Identifying Client Performance and Memory Issues on page 60.

Scalability

Q: If I bring additional projects under configuration management control, what settings should I update?

Additional project data increases configuration management's requirements for memory and cache resources. To determine what adjustments may be required, review the tuning tips provided in Adjusting Integrity Lifecycle Manager server Settings on page 36.

Q: What settings need to be updated if I acquire additional user licenses? Review the information provided under Identifying Server Performance and Memory Issues on page 30.

Q: If I have a proxy that connects to an additional Integrity Lifecycle Manager server, what settings should I update?

You should specify or check the proxy cache settings for each additional repository. You should also verify memory settings for the proxy. For more information, review the information provided in Identifying Proxy Performance and Memory Issues on page 52 and Adjusting Proxy Settings on page 52. Proxy properties are discussed in SI Priming for a Specified Project on page 56 and in Automatic Population of the SI Bulk Data Cache on page 56.

Q: My Integrity Lifecycle Manager client connects to several servers, what performance settings should I check?

You need to specify or check the client cache settings for each additional repository. For more information, review the information provided in Identifying Client Performance and Memory Issues on page 60.

Hardware

Q: I have installed additional memory, what settings should I change to make configuration management run faster?

If configuration management was running smoothly before the additional memory was installed, then no further adjustments are required to improve performance. This is also true if you installed extra memory because of disk swapping issues. If additional memory was installed to address out-of-memory issues, you should increase the server heap size. For more information, see Adjusting Integrity Lifecycle Manager server Settings on page 36.

Q: When should I stop tuning and install additional memory?

Additional memory is required when all of the physical memory is used and you cannot reduce the heap size without encountering swapping problems or error messages.

Q: When should I stop tuning and install a new hard drive?

A new hard drive is required when the bulk data cache uses all of the available disk space and there are frequent collections.

Q: Do I need a faster CPU for the server?

A faster CPU improves performance and reduce the load on the server. Another way to reduce the load on the server is to use the FSA architecture available with Integrity Lifecycle Manager. By using proxies—even on a LAN, loads can be distributed across the proxy cache. Because most operations go through the cache, this is a good method for load balancing.

Cheat Sheet

> **Caution**
>
> This page is a summary. It should only be used after the full document has been read at least once. Failure to do so results in further performance or scalability issues.

Server

Detection

- `server.log*` (Check for Cache Warning and Out Of Memory messages)
- `si/im stats`, `si/im diag --diag=cacheInfo`

Memory (mksservice.conf)

- HeapSize: –`Xmx` and –`Xms`

Database

- Run statistics.
- Check Query times regularly.

Server Bulkdata Caches location: (si.properties/im.properties)

- `si.ServerCache.default.bulkRootDir` (use short dir name)
- `im.ServerCache.default.bulkRootDir` (use short dir name)

Server Cache sizes

- `si.ServerCache.default.size` (allocated in heap)
- `si.ServerCache.default.bulkDiskSize`
- `si.ServerCache.default.bulkMemSize` (allocated in heap)
- `im.ServerCache.default.size` (allocated in heap)
- `im.ServerCache.default.bulkDiskSize`
- `im.ServerCache.default.bulkMemSize`

Proxy-specific

(all server settings apply as well)

Proxy List (is.properties)

Detection

- `si/im stats --target=proxy, si/im diag --diag=cacheInfo --target=proxy`

Proxy Bulkdata Caches location (si.properties/im.properties)

- `si.ProxyCache.default.bulkRootDir` (use short dir name)
- `im.ProxyCache.default.bulkRootDir` (use short dir name)

Proxy Cache sizes

- `si.ProxyCache.default.size` (allocated in heap)
- `si.ProxyCache.default.bulkDiskSize`
- `si.ProxyCache.default.bulkMemSize` (allocated in heap)
- `im.ProxyCache.default.size` (allocated in heap)
- `im.ProxyCache.default.bulkDiskSize`
- `im.ProxyCache.default.bulkMemSize`

> **Note**
>
> Total heap for caches = sum of all proxy caches memory settings + server cache settings

Client

Detection

- `IntegrityClient.log` (Check for Cache Warning and Out Of Memory messages)
- `si/im stats --target=client, si/im diag --diag=cacheInfo --target=client`

Memory: (`IntegrityClient.lax`)

- HeapSize: `-Xmx` and `-Xms`

Client Cache sizes

- `si.ClientCache.default.size` (allocated in heap)
- `im.ClientCache.default.size` (allocated in heap)
- `im.ClientCache.default.bulkDiskSize`
- `im.ClientCache.default.bulkMemSize` (allocated in heap)

Note

Total heap for caches = sum of all memory caches settings * number of SI or IM connections

Links

Databases Resources

- Microsoft SQL Server at http://www.microsoft.com
- Oracle at http://www.oracle.com

Getting Help

PTC Technical Support is focused on delivering the right solutions to issues as they arise. Online support provides easy access to email, web request services, automatic product notifications, and the Integrity Lifecycle Manager Help Center. The Integrity Lifecycle Manager Help Center is a secure database that provides helpful resources such as product documentation, knowledge base articles, product downloads, user forums, presentations, and more. For online support, browse to the Integrity Lifecycle Manager Help Center (http://www.ptc.com/support/integrity.htm).

PTC's global technical support professionals comprise a tightly knit team of problem solvers. This team shares critical information to help you resolve issues in the shortest possible time with optimal results. Support representatives can provide you with various product-related tips and innovative solutions to your unique requirements.