TRIRIGA Wiki Home
Facilities Management ...
Facilities Maintenance
Environmental & Ener...
Real Estate Management
Capital Project Manag...
CAD Integrator-Publis...
IBM TRIRIGA Connect...
IBM TRIRIGA Anywhere
IBM TRIRIGA Applicati...
Support Matrix
Hints and Tips
Installing
Admin Console
Builder Tools
Connector for Busin...
Connector for Esri GIS
Document Manager
Extended Formula
Gantt Scheduler
Globalization
Group Object
Label Manager
Licensing
Object Labels and R...
Offlining
OSLC
Performance
Best Practices for ...
Introduction
Network consider...
System architect...
Operating system...
Database server t...
Database specific...
Application Serve...
IBM TRIRIGA tun...
TRIRIGA Anywhe...
Troubleshooting ...
Information gathe...
Performance Probl...
Performance Analy...
Workflow Analysis ...
IBM TRIRIGA Appli...
Performance Consi...
Understanding you...
Database Indexes f...
SQL Server Index ...
Performance Degra...
DB2 Database Sho...
Platform Logging
Portal and Navigation
Reporting
Reserve
Scheduler Engine (S...
Security
SSO
Styling
System Sizing

You are in: IBM TRIRIGA > IBM TRIRIGA Application Platform > Performance > Best Practices for System Performance > IBM TRIRIGA tuning

IBM TRIRIGA tuning

Like | Updated November 19, 2019 by Jay.Manaloto | Tags: performance, system_performance, tririga, tuning Add or remove tags

Edit Page Actions

Performance | Performance Best Practices | Performance Decision Tree | Performance Analyzer | Workflow Analysis Utility

Best Practices for System Performance.

7 IBM TRIRIGA Tuning

< Back to Table of Contents

Next >

This section discusses settings that can optimize your IBM TRIRIGA environment.

7.1 System Properties

System properties can be used to tune performance in TRIRIGA. For more information on the system properties that can be configured, see the 3.5.3 Installation and Implementation Guide.

7.1.1 TRIRIGAWEB.properties

Most system properties are maintained in the **TRIRIGAWEB.properties** file and are available dynamically in the Administrator Console. For changes in the **TRIRIGAWEB.properties** file to take effect, the application server must be restarted.

The variables and settings in TRIRIGA properties files may change from one version to the next. After completing an upgrade installation, carefully review each newly installed properties file and adjust values as appropriate for your implementation. Some key properties to consider in tuning your system are listed here with their recommended values for ideal performance. For key agent-related properties, see the following section **7.1.2 Agents**.

The following key properties and values can be configured for your environment:

- **CLEAN_HOUR:** Specifies the hour at which the Platform Maintenance Scheduler (Cleanup Agent) starts, in 24-hour time of the server.
  - Recommended value: A time when the system has the least number of users.
  - Default value: 2
- **CLEAN_TIMEOUT:** Specifies the number of minutes that the Platform Maintenance Scheduler (Cleanup Agent) is allowed to run.
  - Default value: 120
- **WF_INSTANCE_SAVE:** Configures when workflow instances should be saved. The primary use of this feature is for debugging in a non-production environment. So, in a production environment, you should always have it set to the recommended value (**ERRORS_ONLY**), because of the significant load on the entire system if turned on. **ERRORS_ONLY** is the default and saves the tracing information only for those workflows that fail with an ERROR condition. The value

- TDI
- Web Graphics
- Workflow
- Release Notes
- Media Library
- Best Practices
- Upgrading
- Troubleshooting
- UX Framework

New Page

Index

Members

Trash

Tags

Find a Tag

analysis application availability_section best_practices cad change_management changes compare compare_revisions customizations customize database db2 exchange find_available_times gantt_chart gantt_scheduler group memory_footprint modifications modify object_label object_revision operating_system oracle performance platform problem_determination reports reserve reserve_performance revision revisioning single_sign-on snapshot space sql_server sso support system system_performance tags: track_customizations tririga troubleshoot tuning upgrade ux version versioning

Cloud | List

Members

of **NEVER** was renamed to **ERRORS_ONLY**, but the system accepts both values, where **NEVER** is equal to **ERRORS_ONLY**. This property can also be changed without a system restart by using the Workflow Agent Manager in the Administrator Console. For more information, see the following sections **7.1.2.e Workflow Agent** and **7.4 Workflow Performance**.
    - Recommended value: **ERRORS_ONLY**
    - Default value: **ERRORS_ONLY**
- **USE_WF_BINARY_LOAD:** If set to **Y**, the system uses the binary workflow load process, which provides a faster load time. The workflow templates are loaded with their stored binary version, if they cannot be found in the workflow template cache.
    - Recommended value: **Y**
- **WF_HISTORY_RETENTION_DAYS:** The Platform Maintenance Scheduler (Cleanup Agent) deletes workflow instances that are not waiting on a user or on approval tasks that are older than this number of days. This value does not have a large impact if the **WF_INSTANCE_SAVE** property is set to **ERRORS_ONLY**. But on a system that is saving a large number of workflow instances, this value can significantly impact system performance.
    - Recommended value: 5
    - Default value: 10
- **DC_HISTORY_RETENTION_DAYS:** The Platform Maintenance Scheduler (Cleanup Agent) deletes completed or obsolete DataConnect jobs that are older than this number of days. If you do not use DataConnect, then changing this value will not affect your system.
    - Recommended value: 5
    - Default value: 10
- **REPORT_MEMORY_USAGE_LIMIT:** Specifies the maximum percentage of available server memory that can be used while running a user report. If a user sees a "There are not enough resources available to run the report" error for a query, it is likely that the query was the cause of the error. However, it also is possible that other concurrent processes could have consumed memory while the query was assembling its results. Valid values are between 0 and 100. Values of 0 and 100 essentially disable any limit being enforced and allow a single query initiated by a single user to run the server out of memory. This value should be tuned so that no report consumes more than 1-2 GB of heap size, but the value can be increased to handle especially large reports.
    - Default value: 90
- **BIRT_MEMORY_USAGE_LIMIT:** Specifies the maximum percentage of available server memory that can be used to assemble the BIRT report query results. If the memory requirement for such a task exceeds the limit, the query will yield an error due to insufficient resources. It is likely that the query was the cause of the error. However, it also is possible that other concurrent processes could have consumed memory while the query was assembling its results. Valid values are between 0 and 100. Values of 0 and 100 essentially disable any limit being enforced and allow a single query initiated by a single user to run the server out of memory.
    - Recommended value: 35
- **TREE_PAGING_SIZE:** Specifies the maximum number of child records to be shown in the hierarchy tree for Location, Organization, Geography, Classification, Cost Code and newly created hierarchical managers. The system includes the child records of the root node in the count. Performance can be affected by increasing this value.
    - Recommended value: 1000
- **BRAVA_EXCLUDE_LIST:** Specifies a list of file extensions that should **not** be viewed with Brava while using the Document Manager component. Any file extension **not** listed is passed to Brava. This property is only used when using Brava. If Brava is not installed, then the property is ignored.
    - Recommended value: **html,htm,svg,rpt,zip,exe,doc,docx,xls,xlsx,ppt,pptx,pdf,txt,xml**
- **SESSION_HISTORY_TRACKING:** Indicates which user sessions are logged to the **SESSION_HISTORY** table. If set to **WEB_USER**, user sessions from IBM TRIRIGA Connector for Business Applications (CBA) are not logged to the **SESSION_HISTORY** table. If your system has a large number of integration transactions, this table can grow very quickly and use system resources.
    - Recommended value: **WEB_USER**
    - Default value: **ALL**
- **BIRT_PROCESS_SERVER...:** The following properties are used to enable BIRT report offloading. BIRT report offloading can eliminate system resource usage by BIRT reports on a server with user sessions by sending the report processing to another JVM. For more information, see the following section **7.3.7 Advanced Reporting (BIRT)**.
    - **BIRT_PROCESS_SERVER_HOST_NAME**
    - **BIRT_PROCESS_SERVER_PORT**
    - **BIRT_PROCESS_SERVER_LISTENING_PORT**

7.1.2 Agents

The IBM TRIRIGA Application Platform utilizes the power of business process **agents** to perform automated work for its applications. When the system identifies an event that requires a business process agent to fulfill, it places the event into a queue where the agent can pull it up and perform work on it. These agents are critical to sustaining a healthy system and managing system performance.

Agents that will **not** be used should be disabled or stopped and should not be configured to start on restart. Not only do these agents consume system resources, but some are multi-threaded, allow for multiple instances, and if not tuned properly, can either restrict or overrun system resources. On a system that has a minimum of 2 servers (JVMs), the majority of agents should be running on a **designated process server** that does not have users logging into it. Agents that require user connectivity, such as the Data Import Agent, are an exception to the rule for running an agent on the process server.

The following topics discuss properties that are related to agents and are managed by editing the **TRIRIGAWEB.properties** file or using the Administrator Console managers. For more information on the agents, see the 3.5.3 Business process agents.

a. Agent Locations and Descriptions

The following table shows the suggested startup locations and descriptions of the agents.

Agent & Location	Description
Data Import Agent Application Server	• Looks for all tab-delimited files that are uploaded and imports the data into the platform. • Should only be turned on if you are using **Data Integrator** (DI). For more information, see the following section **d. Multi-Threaded Agents**.
Report Queue Agent Application Server	• Retrieves queued report requests, processes the report, and notifies the user. • Should only be turned on if you are using **queued reports**. No reports are delivered this way with the as-delivered TRIRIGA. For more information, see the following section **d. Multi-Threaded Agents**.
Object Migration Agent Application Server	• Migrates TRIRIGA objects from one environment to another environment. • Should only be turned on if you are currently performing an **object migration** (OM) task. If on constantly, the system could make unscheduled application changes.
Agent & Location	**Description**
Platform Maintenance Scheduler (Formerly Cleanup Agent) Process Server	• Conducts data cleanup and runs an analysis on the database. Removes all data in the state of null, and removes DataConnect (DC) jobs and staging table entries that are obsolete or completed. Cleans up completed workflow instances that do not have any user-operable tasks (such as user tasks and approval tasks) within the workflow. • Significant agent and should **always** be turned on. Review the server log on a regular basis to confirm its proper execution. For the location of log files, see the section 9.2.1 IBM TRIRIGA Application Platform logs. • The **cleanup** commands in the Administrator Console are different per database type. These can be modified and expanded based on system need. For more information, see the following section **f. Platform Maintenance Scheduler (Cleanup Agent)**.
DataConnect Agent	• Looks for DataConnect (DC) jobs in the Job Control table that are ready to run. When the agent finds a job, it creates an appropriate smart object for the job. Then the

Process Server	agent posts an asynchronous workflow event to initiate the workflow that pulls the external data into the IBM TRIRIGA database tables. • Should only be turned on if you are using **DataConnect**.
Extended Formula Agent Process Server	• Looks for and processes queued extended formulas. • Significant agent and should **always** be turned on.
Formula Recalc Agent Process Server	• Used to recalculate paths. • Should **always** be turned on.
Incoming Mail Agent Process Server	• Downloads mail from a Post Office Protocol 3 (POP3) server or Internet Message Access Protocol (IMAP) server and translates them into email message records. • Should only be turned on if you have an IBM TRIRIGA Connector for Offline Forms license and are using the **Offlining** component.
Object Publish Agent Process Server	• Publishes TRIRIGA objects in the platform. • Should only be turned on when you are making changes to application objects in the **Data Modeler**.
Reserve SMTP Agent Process Server	• Receives and processes reservation emails sent by Microsoft Exchange. This Simple Mail Transfer Protocol (SMTP) receiver service allows Exchange to communicate with TRIRIGA and allows it to manage resources in Reserve. A Microsoft Exchange send connector is configured to forward any email address with the reservation-specific subdomain to the TRIRIGA application server that runs this SMTP agent. • Should only be turned on if you have an IBM TRIRIGA Workplace Reservation Manager (Reserve) license and are using the **Reserve** product.
Scheduler Agent Process Server	• Looks for and processes all scheduled and recurring events in the platform. • Significant agent and should **always** be turned on. For more information, see the following section **d. Multi-Threaded Agents**.
SNMP Agent Process Server	• Receives and processes Simple Network Management Protocol (SNMP) traps (notifications). At a high level, an analytic event from a building equipment asset is created for a SNMP trap. When the SNMP trap is received, a configured workflow is triggered. This workflow is given information on the SNMP trap by using system business objects (triSnmpPdu and triSnmpPduVariable). SNMP traps include those used for the IBM TRIRIGA Real Estate Environmental Sustainability Impact Manager. • Should only be turned on if you are using **SNMP** traps. For more information, see the **10.5.3 Prerequisite setup for using TRIRIGA Real Estate Environmental Sustainability Impact Manager**.
Tivoli Directory Integrator Agent Process Server	• Starts and stops the Tivoli Directory Integrator (TDI) runtime server from within TRIRIGA. Runs on a schedule and monitors the server. If the agent finds the server not running, it attempts to restart it. The TDI server must be running for TDI-based ETL job items to run properly. TDI-based ETL job items include those used to import data for the IBM TRIRIGA Real Estate Environmental Sustainability Impact Manager. • Should only be turned on if you are using **Tivoli Directory Integrator** (TDI) for TRIRIGA as its ETL runtime engine. For more information, see the **3.5.3 Tivoli Directory Integrator Agent**.
Workflow Agent Process Server	• Processes queued workflow events and the asynchronous workflows that are registered for those events. • Significant agent and should **always** be turned on. For more information, see the following sections **d. Multi-Threaded Agents**, **e. Workflow Agent**, **7.4 Workflow Performance**, and **7.5.1.b Limiting the Number of Workflow Agents**.
Workflow Future Agent Process Server	• Processes actions from a workflow that are set up to trigger at a future date. • Lightweight agent and should **always** be turned on. • Used by many applications including, but not limited to, **BIM**, **Job Scheduling**, **Reserve**, and **Lease**.
Workflow Notification Agent Process Server	• Looks for and processes workflow notifications in the platform, including those notifications to be sent at a scheduled time. • Significant agent and should **always** be turned on.

b. Disabling Agents

Disabling agents that are **not** needed is one way to help limit the use of system resources and to keep them being started.

- **AGENTS_NOT_ALLOWED:** This property can be used to limit what agents can be used by administrators.
  - The value is a comma-delimited list of agents that are not allowed to run on this server. For an example, or for the names of the agents to use in this list, see the comments in the **TRIRIGAWEB.properties** file.
  - A blank value allows any agent to be started on a server, but does not start any agent automatically.

c. Multiple JVMs Per Server

If you are planning to have **multiple** JVMs on the same machine, you must set the following properties to allow for unique names per server (JVM) instead of the default host name and ID.

- **INSTANCE_ID:** Overrides the default machine ID.
  - When two or more TRIRIGA servers are running on the same physical machine, **INSTANCE_ID** must be unique for independent agent management.
  - Leave a blank value if you are running a single instance per physical machine.
- **INSTANCE_NAME:** Overrides the default machine name.
  - When two or more TRIRIGA servers are running on the same physical machine, **INSTANCE_NAME** must be unique for independent agent management.
  - Leave a blank value if you are running a single instance per physical machine.

d. Multi-Threaded Agents

The following list shows which agents have their own unique maximum thread count per server. This allows you to set a limit on each agent's individual threads used, therefore allowing you to control their performance. Be aware that increasing the following threads can increase the JVM heap usage, but also increase the JDBC connections and the load on the database server.

These agent threads can also be managed dynamically without a restart by the Threads Manager in the Administrator Console.

- **Data Import Agent:** This multi-threaded agent provides a way to control data loads with the Data Integrator tool. The thread count is also used by IBM TRIRIGA Connector for Business Applications (CBA) to allow for throttling integration requests. The agent does not have to be on to utilize the Connector for Business Applications (CBA).
  - Property: **DataImportAgentMaxThreads**
  - Recommended value is 2 - 8 depending on the need.
- **Report Queue Agent:** This agent retrieves queued report requests then executes the report and notifies user. For more information on queued reports, see the following section **7.3.7.b Queued Reports (Asynchronous)**.
  - Property: **ReportQueueAgentMaxThreads**

  - Recommended value is 1 - 4 depending on the need. If not used, set to 1.
- **Scheduler Agent:** This agent looks for and processes all scheduled and recurring events in the system. This agent should always be on, but the threads needed are low.
  - Property: **SchedulerAgentMaxThreads**
  - Recommended value is 1 - 4 depending on the need.
- **Workflow (WF) Agent:** This multi-server, multi-threaded agent processes queued workflow events and executes the asynchronous workflows registered for them. Setting the Workflow Agent maximum threads to a large value can drastically slow performance of the system. As a general rule, the maximum threads value for the Workflow Agent should not be more than 2 - 4 times the CPU core count on the database server, but can be adjusted higher if tested properly. For more information, see the following sections **e. Workflow Agent** and **7.4 Workflow Performance**.
  - Property: **WFAgentMaxThreads**
  - Related property: **WF_AGENT_MAX_ACTIVE_PER_USER**
  - Recommended value is 4 - 32 depending on the need.
- **CAD Integrator:** This setting is responsible for the server-side work when syncing attached CAD drawings from IBM TRIRIGA CAD Integrator/Publisher. The number of CAD Integrator threads that the platform allocates to IBM TRIRIGA CAD Integrator/Publisher is managed by the Threads Manager in the Administrator Console.
  - Property: **CadIntegratorMaxThreads**
  - Recommended value is 5 depending on the need.

e. Workflow Agent

Notes:

- As a general rule, we do **not** recommend multiple Workflow Agents, unless the additional agents are configured for a **dedicated** users or groups. For more information, see the following section **7.5.1.b Limiting the Number of Workflow Agents**.
- In an environment with **multiple** Workflow Agents:
  - Only add and change the **WF_AGENT_SLEEPTIME** property value at the direction of Support, or if you see contention on the **WF_EVENT** table in the database.
  - If the **WF_AGENT_SLEEPTIME** property value is changed, make sure that each process server running the Workflow Agent has a **distinct** and separate value that is **not** duplicated on any other server that is connected to the same database. This setup can help prevent Workflow Agents from waking up at the same time, and ease contention on the **WF_EVENT** table, by giving more control as to when the Workflow Agent wakes up. On each server, set a different value. But if the property is **not** set or added, the default value is a random number between 4500-5500 milliseconds.
  - The **WF_AGENT_SLEEPTIME** property can be added to the **TRIRIGAWEB.properties** file to change the time when the Workflow Agent wakes up and checks for new events. If there are events in the queue already, the **WF_AGENT_WAITTIME** property is used to wait before picking up new workflows that are already in the queue.

The **Workflow (WF) Agent** is one of the most vital agents and should have at least one instance running at all times to process asynchronous workflows. Almost every application process in TRIRIGA utilizes synchronous and asynchronous workflows, including all types of integrations, especially TRIRIGA CAD Integrator/Publisher. The user's perceived performance of the system can be affected by the Workflow Agent if it is not processing asynchronous processes quickly enough. If the Workflow Agent properties are not tuned properly, then the workflow queue can get backed up and grow quickly.

The Administrator Console allows you to set up and manage all the workflow tuning properties listed here.

There are multiple ways to manage and improve Workflow Agent performance:

- **Multiple Workflow Agents:** If multiple agents are set up, you can designate specific users for a Workflow Agent exclusively or to give priority. As a general rule, we do **not** recommend multiple Workflow Agents, unless the additional agents are configured for a **dedicated** user. For more information, see the following section **7.5.1.b Limiting the Number of Workflow Agents**.
- **Tune the Threads and Threads Per User:** With multiple agents, you can configure threads independently from each other. A typical scenario for having multiple agents would be when you have one process server setup as a general process server, and another process server setup for integrations or data loads with a **dedicated** user to process these items.
  - Property: **WFAgentMaxThreads**
  - Property: **WF_AGENT_MAX_ACTIVE_PER_USER**. If you have a Workflow Agent that is assigned to a single user, for example, a unique user for integration purposes, you should set this value equal to the value of **WFAgentMaxThreads**. The workflow max active per user property should be set to 70% of the agent maximum threads for general purpose process servers. This prevents a single user from maxing out the use of workflow threads.
- **Workflow Instance Saving:** The workflow instance recording property **WF_INSTANCE_SAVE** should be set to **ERRORS_ONLY** to achieve the best workflow performance. For more information, see the following section **7.4 Workflow Performance**.

f. Platform Maintenance Scheduler (Cleanup Agent)

The **Platform Maintenance Scheduler** (formerly Cleanup Agent) performs multiple system cleanup activities at a specified time each day. The Platform Maintenance Scheduler is a critical agent in sustaining your TRIRIGA system. This agent should always be on to help maintain system health. By default, the Platform Maintenance Scheduler performs the following activities:

- Clean up BO instance records that are stale, such as records marked for delete (null) and temporary objects.
- Clean up workflow instance data.
- Clean up scheduled events.
- Clean up Document Manager data.
- Execute cleanup commands (such as database analyze).

Also mentioned in the previous section **7.1.1 TRIRIGAWEB.properties**, the following **cleanup** commands are maintained in the **TRIRIGAWEB.properties** file, are configurable with the Platform Maintenance Scheduler Manager in the Administrator Console, and are the only components that can be turned off if you choose to run them externally from the agent.

- **CLEAN_HOUR:** Specifies the hour at which the Platform Maintenance Scheduler (Cleanup Agent) starts, in 24-hour time of the server.
  - Recommended value: A time when the system has the least number of users.
  - Default value: 2
- **CLEAN_TIMEOUT:** Specifies the number of minutes that the Platform Maintenance Scheduler (Cleanup Agent) is allowed to run.
  - Default value: 120
- **WF_HISTORY_RETENTION_DAYS:** The Platform Maintenance Scheduler (Cleanup Agent) deletes workflow instances that are not waiting on a user or on approval tasks that are older than this number of days. This value does not have a large impact if the **WF_INSTANCE_SAVE** property is set to **ERRORS_ONLY**. But on a system that is saving a large number of workflow instances, this value can significantly impact system performance.
  - Recommended value: 5
  - Default value: 10
- **DC_HISTORY_RETENTION_DAYS:** The Platform Maintenance Scheduler (Cleanup Agent) deletes completed or obsolete DataConnect jobs that are older than this number of days. If you do not use DataConnect, then changing this value will not affect your system.
  - Recommended value: 5
  - Default value: 10

g. Extended Formula Agent

Ensure that records are **not** being put into the Extended Formula queue when it is **not** required. Objects that are being loaded should only have their formulas calculated once at the end of the process. If you encounter any out-of-the-box workflows that are not exhibiting this behavior, please contact IBM Support.

h. Incoming Mail Agent

Invalid email addresses in the TO: or CC: line can impact the acceptance and delivery of the individual email messages. The **Incoming Mail Agent** will attempt to process all mails sent to the inbox. If an email contains an invalid address, that message may be removed from the inbox, **not** processed, and the agent will continue processing the other messages in the inbox.

For outgoing mails, an invalid email in the TO: or CC: line may cause that individual message to **not** be sent, and an error will be thrown in the log. Other outgoing notifications in the queue will continue to be processed.

7.2 Reserve Performance

A **reserve** (reservation) query is a query that searches all available spaces in an application. The query response time can vary greatly depending on the data composition and setup of the application. You can apply several best practices to significantly increase performance of reserve queries within your Reserve implementation.

For additional information regarding the Reserve application, see the Reserve section of this IBM TRIRIGA wiki.

7.2.1 Filtering

Filtering is the most important application design to consider with any Reserve application. When a reserve query is executed, filters should be applied to ensure the query searches against the smallest result set possible. Fewer records in the result set means the reserve query needs to check the availability of fewer records.

For example, floor **pagination** and floor **filtering** could be added to a building to increase performance. If a reserve query is changed to search a floor instead of the entire building, it will dramatically improve performance if those spaces are spread out across many floors. Changing the application design to enforce a user to pick a floor before performing a search, and providing a "next floor" option, is a great alternative to significantly reduce the reserve query time. Filtering of all kinds should be applied at every possible instance, to ensure the smallest result set is being checked for availability by the reserve query.

7.2.2 Empty Filter

In many applications, a reserve query is executed when the form initially loads and immediate results are returned. This behavior impacts the form load time. This behavior can be eliminated by setting up the application to filter against a building that has **zero** spaces. A "building" can be created called <Please select a building>, and this building can be added to the form and set as the **default** building when the form loads. This filter results in significantly faster form loading times. When the user changes the building selection, the query then searches for available spaces, removing the execution of the query when the form loads.

7.2.3 Enable Directed Search

Starting in IBM TRIRIGA Application Platform 3.4.1, a **direct search** feature enables a reserve query to stop processing and return results as soon as a set number of records are found using the quantity set in the **triAvailabilityResultsLimitNU** numeric field. This feature can be enabled by adding the **triAvailabilityResultsLimitNU** field to any object and form where the reserve query is being run. The platform looks for this field and uses the value to limit the results.

For example, if a site that is being searched has 1000 available spaces, then the **triAvailabilityResultsLimitNU** field can be set to a default of 25. The search stops processing when the first 25 results are found. This feature is a great way to allow a user to search a large results set and maintain performance. If the field is set to 0, then the query returns all results.

This field can also be incorporated into the application design to enable a user to specify the number of records they want to be returned. For consistent searches, the field can also be hidden with a default value. From a usability perspective, the user sees immediate results and the reserve query does not process any longer than necessary.

7.2.4 Enable Multi-Threading

Starting in IBM TRIRIGA Application Platform 3.4.1, a **multi-threading** feature enables a system administrator to specify the total number of threads that are available to the reserve query, and how many threads each reserve query can consume. These properties can be tuned based on how many concurrent reservations are expected to be executed at one time.

The following properties can be added to the **TRIRIGAWEB.properties** file:

- **ENABLE_CONCURRENT_AVAILABILITY:** If set to **true**, this feature is enabled.
- **CONCURRENT_AVAILABILITY_POOL_SIZE:** Specifies the total number of threads that are available to the reserve query.
- **CONCURRENT_AVAILABILITY_REQUEST_BATCH_SIZE:** Specifies how many threads each reserve query can consume.

Here are some example values from the **TRIRIGAWEB.properties** file:

```
# Determines whether process Reserve Queries concurrently.
#
ENABLE_CONCURRENT_AVAILABILITY=true

# If ENABLE_CONCURRENT_AVAILABILITY is true, this is the maximum number of system wide
# threads that will be used in processing availability.
#
CONCURRENT_AVAILABILITY_POOL_SIZE=200

# If ENABLE_CONCURRENT_AVAILABILITY is true, this is the maximum number of threads each
# availability request can use to process availability.
#
CONCURRENT_AVAILABILITY_REQUEST_BATCH_SIZE=10
```

7.2.5 Debugging Reserve

To **debug** the performance of directed search and multi-threading, the following categories can be added as manual categories, at the bottom of the Platform Logging Manager in the Administration Console.

```
com.tririga.platform.concurrent
```

```
com.tririga.platform.query.reserve
```

7.2.6 Performance Indexes

Recommended indexes that significantly improve Reserve performance in IBM test environments can be accessed as follows:

- 5.3.3 Reserve Indexes for DB2
- 5.4.3 Reserve Indexes for Oracle
- 5.5.3 Reserve Indexes for SQL Server

7.3 Query and Report Performance

There are multiple solutions for accessing and presenting the data maintained by the TRIRIGA system. Simple tabular reports, queries, and graphs that combine data from multiple records into a single presentation can be produced using the **Report Manager**. If you need a multiple record presentation that is beyond the capabilities of the Report Manager, you can use the **Advanced Reporting** feature. TRIRIGA uses Eclipse Business Intelligence Reporting Tools (BIRT) as the enabling technology for the Advanced Reporting feature. For more information on reporting for TRIRIGA, see the 3.5.3 Reporting User Guide.

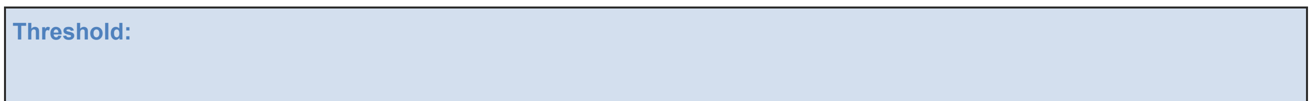
When isolated application processes seem slow, a performance log analysis should be conducted. If slow running queries are identified as the primary bottleneck, the following recommendations can help with tuning. For more information, see the Performance Analyzer or the presentation on IBM TRIRIGA Application Platform 3.x Performance Log Analysis.

7.3.1 Query Tuning

Review custom **queries** and **reports** for efficient SQL and use of indexes. In all of these cases, improving the efficiency of user queries also improves the efficiency of the workflows that use them.

Many long running queries and reports are typically found to be those that are not properly filtered. In addition, most of the filters in reports and queries are generated by individual users with run-time filter operators and by association filter operators defined in the Report Manager. While these are powerful features of TRIRIGA, they can potentially produce inefficient SQL.

In the TRIRIGA Administrator Console, the **Platform Logging** page provides a way to turn on debug level SQL query information. This can be utilized to determine the origin of a query from the TRIRIGA platform.



Threshold:

- Any query that takes more than **100 ms** is a likely candidate for review. Verify the following tips and make changes wherever possible. For more ways to track performance, see the following section **7.3.6 Using Report Run History to Track Query Performance**.

a. Important Tips for Creating Well-Tuned SQL Queries

- **IN vs. NOT IN:** Using "NOT IN" or "!=" will result in a full table scan. Replace with "IN" or "NOT EXISTS".
- **Wildcard Character:** Eliminate the use of the leading % wildcard character. Using this character will result in a full table scan.
- **Index Columns:** Order index columns so that the number of rows returned will be reduced earlier. For instance, using a Boolean column type in the first position of the index would only be able to eliminate about half the rows, while using a column such as SPEC_ID in the first position would give you a much smaller result set.
- **Reverse Association:** Having the reverse association as "No" provides a significant performance improvement over having it as "Yes". If it is required to be "Yes", then understand why it is needed to be a "Yes". For more information, see the following section **7.3.4 Reverse Association Queries**.
- **Association Filter:** Using a separate association per module or per business object is better than against all modules or all business objects.
- **Association Filter Queries:** Check if the query can be changed to a multiple business object query using an association. This is faster than using association filter queries.
- **Multiple Business Object Queries vs. Direct Smart Section**: There could be a difference depending on a number of factors on when to use a multi-BO query versus directly choosing the smart section fields. It is a good idea to try queries either way; either by using the smart section directly in the field chooser, or by adding the business object using the exact association as a multiple-business-object query.
- **Nested Query Optimization:** If a query has more than two nested levels, analyze the requirements and determine if the levels can be reduced by adding filters to reduce the number of hops.
- **System Variables Correctly Used:** Make sure that system variables such as $$RECORDID$$, $$USERID$$, and $$ORGANIZATION$$ are correctly used.
- **Portal Section, Manager, Master Detail, and Find Queries:** Select the "Prompt Before Query" option in the query definition for large data sets. Avoid returning all records with all statuses. Instead, use as many design-time filters as possible to avoid full table scans.
- **Sorting:** Reduce the number of sorting elements to reduce the "cost" of the query. For more information, see the following section **7.3.5 Query Sort Order**.
- **Where Used:** Using the "Where Used" tab in queries helps to determine the places it impacts before the change.

7.3.2 Home Portal and Other Portals

The **Home** portal is a heavily-used component of TRIRIGA, often with many sections containing reports and queries. As such, it is important to ensure that any portal section queries are well-tuned to avoid performance issues. In general, all pre-defined queries must be well-tuned for optimal performance across the system.

A large number of portal sections on the **Home** portal or any other portal can contribute to slow performance. In the portal, set the portal section to "**collapsed by default**" whenever possible, so a limited number of sections are open on any portal. Portals that are set to "**auto refresh**" can also cause slow performance.

7.3.3 Fields and Extended Formulas

Threshold:

- Any extended formula that takes more than **100 ms** is a likely candidate for review. View the formula for improvements. Association queries may take a longer time and hence take a longer time for calculations. Note that, at times, improving the association queries increases the performance calculations.

Review the following checklist to optimize the performance of fields and extended formulas:

- **Association Queries:** Check the association queries used in the formula. See if the queries can be improved.
- **Regular Formula:** See if it is possible to change the formula to a regular formula.
- **Field Requirements:** Verify if the field with the extended formula is required for the functionality of the business object or form. Extraneous extended formula calculations can be introduced when a field is copied from a business object in which the extended formula calculation is used into a place where the result is not needed. For example, straight-line calculations are required in lease abstract but may not be required for the field (triRentStraightLineNU).
- **Associated Data:** Consider using the query on an extended formula to get the associated data instead of using the field type to get the data being associated. There is considerable improvement in using a query instead of a field.
- **Sorting:** Avoid sorting on the query when used for an extended formula. For more information, see the following section **7.3.5 Query Sort Order**.
- **Where Used:** Using the "Where Used" information on fields helps to determine whether this field is used on a form.

7.3.4 Reverse Association Queries

For performance reasons, the **Reverse Association** flag is being deprecated and is planned for removal in an upcoming release.

To begin this transition, a new property was added to **TRIRIGAWEB.properties** named **ALLOW_REVERSE_ASSOCIATION**. The default for this setting is currently **TRUE**, allowing the legacy behavior of reverse associations to be run in reports and queries as defined in Report Manager. If you want to disable reverse associations from being run for testing purposes, you can set **ALLOW_REVERSE_ASSOCIATION=FALSE** and no queries will be executed with reverse associations. This should be performed on a test environment first to ensure the integrity of data coming back from queries. In most cases, the flag is not necessary, and in cases that it is, some minor metadata changes can be made to render it unnecessary.

For any queries or reports defined in the Report Manager with the Association Filter set with the **Reverse Association** flag (either "Yes" or "No"), the **ALLOW_REVERSE_ASSOCIATION=FALSE** setting ignores this flag, and interprets and runs queries as if this flag was not set, so only forward associations are returned.

The query that is generated to return both the forward and reverse association is expensive from a database perspective. It can lead to poorly performing queries that can consume significant CPU across the database server, which can slow down other queries and operations on the database when they are executed. In most cases, it adds unnecessary overhead without affecting the query results. The platform has not allowed new queries to be created with the **Reverse Association** flag since 3.4.2, nor allowed queries to be copied with the flag since 3.5.0.

If you want to run your system with this property set to **FALSE** to evaluate the performance implications, we strongly advise that you do so in a dev/test environment first. You may encounter a query on a manager, query section, or workflow that no longer has the results you were expecting. If this happens, you should first examine the **Association** tab of the records in question and note the correct forward association string. Then you can open the query in Report Builder, go to the Association Filter, and examine the association string used there. If they are different, then update the report to use the correct association string. If your records only have a one-way association, you will need to create the forward and reverse association in Association Manager, and then update your data via workflow to set the forward and reverse association.

7.3.5 Query Sort Order

Case-insensitive sorting on query results is known to cause performance slowdown when processing large volume of data under stress. Starting in IBM TRIRIGA Application Platform 3.5.3, you can turn off the forced case-insensitive **Order By** on query results by using the following property and setting in the **TRIRIGAWEB.properties** file:

- **REPORT_CASE_SENSITIVE=NATIVE_DB_CASE_SORT**

The application server must be restarted for changes in the **TRIRIGAWEB.properties** file to take effect. Turning off the case-insensitive **Order By** might result in improved database performance. But the sort results might be ordered differently as follows.

Here are examples of **DB2** case-insensitive versus case-sensitive sorting:

- **REPORT_CASE_SENSITIVE=FORCE_CASE_INSENSITIVE:** Case-insensitive sorting.
  ```
  SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
  T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TEST T1 WHERE
  T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY UPPER(T1.NameTX)
  ```
- **REPORT_CASE_SENSITIVE=NATIVE_DB_CASE_SORT:** Native database case-sensitive sorting.
  ```
  SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
  T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TEST T1 WHERE
  T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY T1.NameTX
  ```

Here are examples where **SQL Server** case-insensitive and case-sensitive sorting give the **same** result:

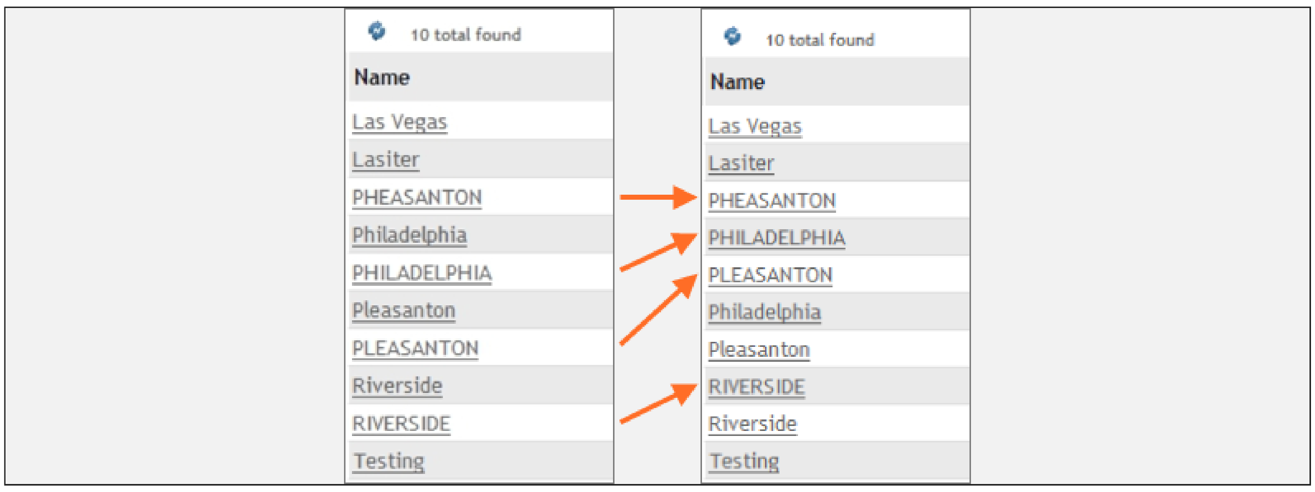
- **REPORT_CASE_SENSITIVE=FORCE_CASE_INSENSITIVE:** Case-insensitive sorting.

  ```
  SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
  T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TEST T1 WHERE
  T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY UPPER(T1.NameTX)
  ```

- **REPORT_CASE_SENSITIVE=NATIVE_DB_CASE_SORT:** Native database case-sensitive sorting.

  ```
  SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
  T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TEST T1 WHERE
  T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY T1.NameTX
  ```

Example of DB2 Sort: Case Insensitive vs. Case Sensitive.



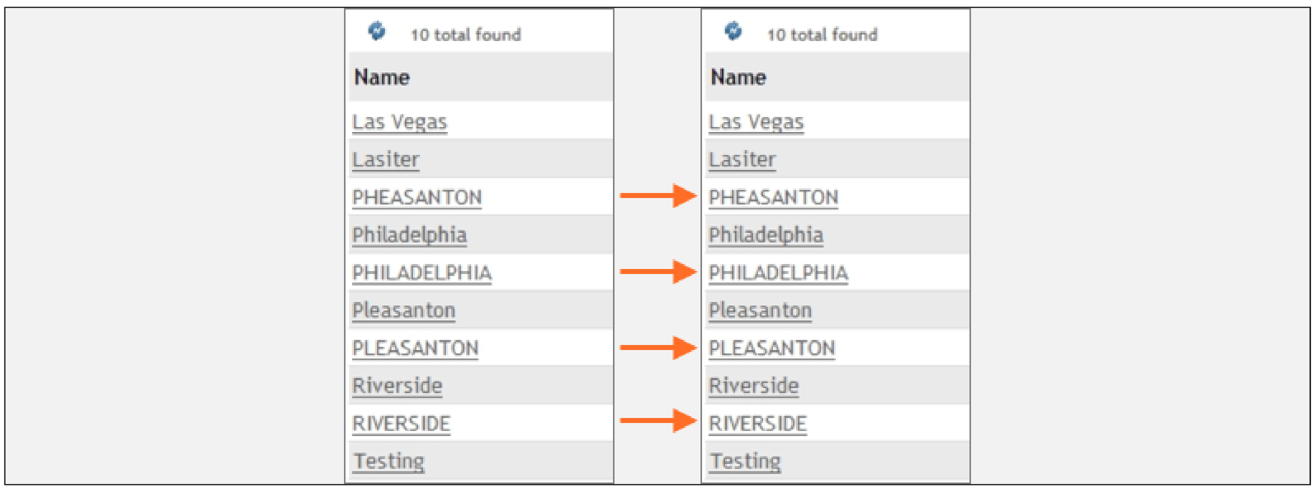
10 total found

Name
Las Vegas
Lasiter
PHEASANTON
Philadelphia
PHILADELPHIA
Pleasanton
PLEASANTON
Riverside
RIVERSIDE
Testing

10 total found

Name
Las Vegas
Lasiter
PHEASANTON
PHILADELPHIA
PLEASANTON
Philadelphia
Pleasanton
RIVERSIDE
Riverside
Testing

Example of SQL Server Sort: Case Insensitive vs. Case Sensitive.



10 total found

Name
Las Vegas
Lasiter
PHEASANTON
Philadelphia
PHILADELPHIA
Pleasanton
PLEASANTON
Riverside
RIVERSIDE
Testing

10 total found

Name
Las Vegas
Lasiter
PHEASANTON
Philadelphia
PHILADELPHIA
Pleasanton
PLEASANTON
Riverside
RIVERSIDE
Testing

7.3.6 Using Report Run History to Track Query Performance

In the IBM TRIRIGA Application Platform 3.5.0 release, a new feature was added that allows you to track the performance and monitor who runs queries and reports in Report Manager. A flag can be set on the report definition that will track who has run the report, and the time it took to retrieve the information from the database.

To enable the report run history, open the report in the Report Manager, click the **General** tab, and select **Track History**.

Notes:

- Enabling this tracking causes just a slight overhead, but it should still be used with **caution**.
- After tracking is enabled, you will need to let your users use the system for a while so that some data can be saved. After a few days, you can then start to analyze what is happening in your system.

The data can answer questions such as the following:

- Which reports have run the longest overall?
- Of the reports that have run, which have the average longest runtime?
- For the users in the system, who runs the most reports?
- Of the users in the system, who have run the longest reports?

- **Which reports have run the longest overall?**

  ```
  SELECT MAX(duration), rth.rep_name
  FROM REGRESSION35X.rep_history rh, REGRESSION35X.REP_TEMPLATE_HDR rth
  WHERE rth.REP_TEMPLATE_ID = rh.REP_TEMPLATE_ID
  GROUP BY rth.rep_name
  ORDER BY MAX(duration) DESC;
  ```

- **Of the reports that have run, which have the average longest runtime?**

  ```
  SELECT AVG(duration), rth.rep_name
  FROM REGRESSION35X.rep_history rh, REGRESSION35X.REP_TEMPLATE_HDR rth
  WHERE rth.REP_TEMPLATE_ID = rh.REP_TEMPLATE_ID
  GROUP BY rth.rep_name
  ORDER BY AVG(duration) DESC;
  ```

- **For the users in the system, who runs the most reports?**

  ```
  SELECT COUNT(*), uc.user_account
  FROM REGRESSION35X.rep_history rh, REGRESSION35X.user_credentials uc
  WHERE rh.user_id = uc.user_id
  GROUP BY uc.user_account
  ORDER BY COUNT(*) DESC;
  ```

- **Of the users in the system, who have run the longest reports?**

  ```
  SELECT MAX(duration), uc.user_account
  FROM REGRESSION35X.rep_history rh, REGRESSION35X.user_credentials uc
  WHERE rh.user_id = uc.user_id
  GROUP BY uc.user_account
  ORDER BY MAX(duration) DESC;
  ```

7.3.7 Advanced Reporting (BIRT)

The same best practice principles suggested for reporting should be followed when designing and building **BIRT reports**. The following optional features of the TRIRIGA platform are used to improve the performance of the front-end application server.

a. BIRT Report Offloading

You can specify BIRT reports to be **offloaded** and processed by a separate TRIRIGA server that is configured as the **BIRT process server**. If you do not configure a separate BIRT process server, the BIRT report processing is done locally.

The following system properties are used to enable BIRT report offloading. BIRT report offloading can eliminate system resource usage by BIRT reports on a server with user sessions, by sending the report processing to another JVM. The offloaded server is simply another TRIRIGA server that can be dedicated to this task or be the process server already set up for back-end agents.

- **BIRT_PROCESS_SERVER_HOST_NAME**
- **BIRT_PROCESS_SERVER_PORT**
- **BIRT_PROCESS_SERVER_LISTENING_PORT**

For more information, see **Configuring BIRT process servers** or the **3.5.3 Reporting User Guide**.

b. Queued Reports (Asynchronous)

To execute a BIRT report **asynchronously** and notify the user when the report is ready to view, you can mark a report as **queued**. Consider running reports that take more than one minute to run asynchronously instead of synchronously. Running a report asynchronously lets the report run on the server when the server is not busy with other tasks. If a queued report has runtime filters or BIRT filters, they are presented to the user before the report is queued, as they are for a non-queued report.

You **can** launch from and access queued BIRT reports from the following places: My Reports, Report Manager, manager query lists, and query sections. You **cannot** launch from and access queued BIRT reports from the following places: portal sections and form actions. But if a queued report is attached to a portal section, the system launches it in a non-queued manner.

When a queued report finishes processing, the system sends a notification to the user’s Notifications portal section. To access the report, the user clicks the hyperlinked subject line. A queued BIRT report attached to a workflow notification executes along with the notification. However, if the report contains filters that do **not** have a default value, the report generation may **error** out because there is no user to provide the values. You need to set the default values for all filters when you define such a report.

c. Maximum Available Server Memory

You can change the maximum percentage of available server memory that can be used to create the BIRT report query results. To do so, use the system property **BIRT_MEMORY_USAGE_LIMIT**. For information on using this property, see the previous section **7.1.1 TRIRIGAWEB.properties**.

7.3.8 Geography-Organization Security Check

You can choose an **alternative** way to run a Geography and Organization **security check** on a query. The generated SQL is less expensive and might perform more efficiently depending on your Geo/Org hierarchy structure. To use this alternative method, you must comply with the following limitations:

- The **triPathSY** field must be **part** of the Organization and of all Geography business objects.
- The values of **triPathSY** fields on Organizations and all Geography records must be **consistent** with the hierarchy.
- The Organization and Geography used within any security group must contain **fewer** characters than the size of **triPathSY** on the Organization and Geography business objects.

This alternative method uses the **GEO_ORG_SECURITY_CONTRIBUTE_BEHAVIOR** property in the **TRIRIGAWEB.properties** file. By default, this property is set to **COALESCE** as the legacy way of using the **IBS_SPEC_STRUCTURE** along with recursive subselects to contribute Geo/Org security. However, setting this property to **PATH** offers the alternative method by contributing simpler SQL that runs a compare against the **triPathSY** field, which is why **triPathSY** must be consistent (in sync) with the **IBS_SPEC_STRUCTURE** to ensure proper Geo/Org security.

You can determine whether your Geography and Organization hierarchies are **in sync** with their **triPathSY** values, by using the Database Manager in the Administrator Console. In the Database Manager > Database Admin Tasks, select **Check Organization Hierarchy** or **Check Geography Hierarchy**. Read the popup and click **OK** to confirm. This check process might take a long time to run depending on the size of your Geo/Org hierarchy. You can monitor the **server.log** for feedback regarding any out-of-sync entries and process completion.

7.4 Workflow Performance

Workflows are vital and one of the key components of the architecture of TRIRIGA. When the focus of performance tuning is on the applications, then the workflows for the process in question should first be benchmarked and analyzed for tuning. This can be done by utilizing many methods including, but not limited to, platform logging features and performance log analysis.

For more information, see the **Workflow Analysis Utility**, the Performance Analyzer, or the presentation on IBM TRIRIGA Application Platform 3.x Performance Log Analysis.

7.4.1 Workflow Instance Data

Workflow instance recording saves the step-by-step execution information to the database. This data also shows the path taken in a graphical display, including links in task details and information on nested workflows.

Notes:

- The primary use of workflow instance recording is for **debugging** in a non-production environment.
- Do **not** use workflow instance recording in a production environment. It causes **significant** overhead on the system and should **never** be used in a production environment. It can increase workflow response time by 50% or higher and writes significant amounts of data to the database.
- Do **not** use workflow instance recording when doing data loads or performing batch processing. The saving of instances will not only **slow down** the processing, but will also consume a **huge** amount of database space, and will **greatly** impact the Platform Maintenance Scheduler cleanup.

Also mentioned in the previous section **7.1.1 TRIRIGAWEB.properties**, the following properties and values can be configured:

- **WF_INSTANCE_SAVE:** Configures when workflow instances should be saved. The primary use of this feature is for debugging in a non-production environment. So, in a production environment, you should always have it set to the recommended value (**ERRORS_ONLY**), because of the significant load on the entire system if turned on. **ERRORS_ONLY** is the default and saves the tracing information only for those workflows that fail with an ERROR condition. The value of **NEVER** was renamed to **ERRORS_ONLY**, but the system accepts both values, where **NEVER** is equal to **ERRORS_ONLY**. This property can also be changed without a system restart by using the Workflow Agent Manager in the Administrator Console.
  - Recommended value: **ERRORS_ONLY**
  - Default value: **ERRORS_ONLY**
- **USE_WF_BINARY_LOAD:** If set to **Y**, the system uses the binary workflow load process, which provides a faster load time. The workflow templates are loaded with their stored binary version, if they cannot be found in the workflow template cache.
  - Recommended value: **Y**

7.4.2 Workflow Optimization

Threshold:

- Any **synchronous** workflow that takes more than **500 ms** is a likely candidate for review. Based on your user requirements, check to see if the workflow can be run as **asynchronous**. Make sure that a workflow does not require synchronous processing. Otherwise, problems can occur.
- Any **asynchronous** workflow that takes more than **2 seconds** is a likely candidate for review.

Review the following checklist to optimize the performance of workflows:

- **Transaction Control and Multi-Record Processing:** The **Create Record** and **Modify Records** tasks can cause a large number of database commits, but they also can cause excessive heap usage. The specific situation depends on the records, data, and number of records that are being processed. On the one hand, if you reduce the number of commits by **increasing** the number of records per commit, this can increase performance by reducing the load on the database. However, on the other hand, if you attempt a **large** set of records (e.g. greater than 10K records) at one time, this can cause excessive heap usage because the task will **not** commit until **all** of the records are created or modified. To determine the **best balance**, use the following methods:
  - **Record Sets:** Use record sets whenever possible to minimize the number of database commits.
  - **Transaction Section:** Use the task's Transaction section when the record set is large or unknown, to commit every N records (e.g. every 100 records) to prevent excessive heap usage.
- **Query Task vs. Retrieve Records Task:**
  - Use the **Query** task when: (1) Associated records are used as filters, (2) Getting general information, such as open orders or occupied spaces, and (3) Intermediate record set might be large.

    - Use the **Retrieve Records** task when: (1) Getting records with the largest or smallest value, (2) Accessing financial records with a token, and (3) Filtering an existing record set (the result of another step).
- **End Task vs. Stop Task:**
    - **Stop Task:** A **Stop** task is used to indicate the abnormal and immediate stopping of a workflow. When a workflow reaches a Stop task, no more of its tasks will be performed. The workflow is considered to have completed abnormally.
    - **End Task:** An **End** task is used to indicate the end of a workflow. When a workflow reaches an End task, the workflow is complete and there are no more tasks to be performed. The workflow is considered to have completed normally, so saving of instances will follow the server's or the Start task's instance save.
    - When developing workflows, use **End** tasks for most instances when you want to exit the workflow gracefully, and all work is done. You would only use the **Stop** task in worst-case scenarios, as this will cause the entire workflow instance stack to be saved. This would not be desirable, especially in data-load scenarios, since the saving of workflow instances is extremely expensive. When in doubt, use the **End** task.
- **Query Task Filter:** If possible, when filtering records in a Query task using the list comparison, use the "Associated To" option to reduce the tasks in a workflow.
- **Associate Records Task:** Use the **Associate Records** task for the forward association only. The platform does the reverse association automatically if the association is defined in the Association Manager.
    - However, for performance reasons, the **Reverse Association** flag is being deprecated in queries. For more information, see the previous section **7.3.4 Reverse Association Queries**.
- **Switch Task:** Use the **Switch** task to call a workflow only when needed. For example, in a Permanent Save Validation workflow.
- **Extended Formula:** Trigger an extended formula only when needed ("Formula" property). For more information, see the previous section **7.3.3 Fields and Extended Formulas**.
- **In Memory Smart Object:** If possible, use the In Memory Smart Object for helper records that do not need to be saved.
- **Cascade Read-Only:** Use the "Cascade Read-Only" option to eliminate the need to trigger separate workflows to make the child record read-only. The "Cascade Read-Only" flag is set in the Association Manager.
- **Workflow Analysis Utility:** Identify any workflow "hot spots" by using the Workflow Analysis Utility.

7.4.3 Workflow Analysis Utility

The **Workflow Analysis Utility** analyzes workflow performance and process execution. The utility reads IBM TRIRIGA performance logs and displays performance analytics for workflows, including workflow execution time (how long different workflows take to run), and process flow (the order in which workflows ran and what triggered them to run).

For information on downloading this utility, see the Workflow Analysis Utility. In addition, workflow analysis can be performed by using the steps in the presentation on IBM TRIRIGA Application Platform 3.x Performance Log Analysis.

7.5 Integration Framework

The IBM TRIRIGA Application Platform provides a variety of mechanisms to **integrate** with external applications. These integration options are used to integrate TRIRIGA with other systems. The main integration components are as follows:

- Connector for Business Applications (CBA).
- DataConnect (DC).
- Data Integrator (DI).
- Integration Object (IO).
- Open Services for Lifecycle Collaboration (OSLC). For more information, see the OSLC Getting Started Guide.

For more information on the integration components, see the 3.5.3 Integrating data with external applications.

These integration components can significantly affect the performance of the system, and should be taken into consideration when sizing and tuning a system. The incorporation of both **multi-threading** and **clustering** greatly increases throughput in an integrated environment. The TRIRIGA platform employs a multi-threaded integration model. The Connector for Business Applications (CBA) model is set up to process multiple inbound transactions at once. In a large and integrated system, you must tune the application and the overall environment parameters, including the parameters for the application server.

Other recommendations to consider when tuning integrations for performance are as follows:

- **Dedicated Server:** If implementing a large number of integrations, use a dedicated server for integrations.
- **Off-Peak Scheduling:** Schedule regular integration jobs during off-peak user times.
- **Session Tracking:** Turn off the session tracking for integrations.
- **Workflow Execution:** Minimize the number of workflows executed by integration processes.
- **Data Loading:** For large data loads, migrations and batch jobs: (1) Notify users, (2) Schedule during a maintenance window and lock out users, and (3) Configure the system for optimal data loading, for example, maximize JVM heap, increase workflow agents and threads, and enable only essential components.

7.5.1 Integration Workflow Processing and Performance

All of the integration components mentioned here use the workflow engine, and therefore, can also be tuned by following the best practice guidelines in the previous sections **7.1.2.e Workflow Agent** and **7.4 Workflow Performance**.

a. Disabling Workflows Unused by an Operation

When TRIRIGA performs **integration operations**, such as importing large numbers of floor and space records via DataConnect (DC), you can increase the throughput by analyzing which workflows are executed as part of the process, and **disabling** those workflows that are **not** needed by the nature of the operation (such as using DC to create many records with known good data).

One way to find out which workflows are executed is to enable all **workflow instances** to be recorded and then inspect them after going through the process once. However, this can be very tedious and easy to miss many workflows that are executed as side-effects of creating and associating records. In addition, enabling workflow instance recording can cause significant overhead to the process and entire system, so try to avoid it except when debugging a part of the processing. In general, you want all instance recording disabled. You can set the **WF_INSTANCE_SAVE** option in **TRIRIGAWEB.properties** and temporarily override it if needed by using the Workflow Agent Manager in the Administration Console. For more information, see the previous section **7.4.1 Workflow Instance Data**.

Instead of workflow instance recording, if you want to know which workflows are executed and why, you can enable the workflow **performance logging** (both synchronous and asynchronous) and collect the **performance.log** file from the process server that is running the Workflow Agent. Then you can process and analyze the **performance.log** by using the **Workflow Analysis Utility** to get an interactive view into what is running, why, and how long. For more information, see the Workflow Analysis Utility.

Also, if the integration processing is done on a regular basis, consider using the **Integration** flag in the workflow start conditions to skip workflows instead of temporarily retiring them.

b. Limiting the Number of Workflow Agents

Notes:

- As a general rule, we do **not** recommend multiple Workflow Agents, unless the additional agents are configured for a **dedicated** user.

The point of running **multiple** workflow servers is to allow workflow processing to be done so that it is **fair** to all users, **not** necessarily to **increase** the throughput of the number of workflows processed. Adding multiple Workflow Agents to a single environment can slow down processing and cause undesirable results, if workflows are not written with **multi-threading** in mind.

The optimal approach is to assign **secondary** Workflow Agents to **dedicated** power users (for example, integrations) that tend to run more workflows than a typical user. This can be done with the Workflow Agents Manager in the Administration Console.

If multiple Workflow Agents are left open (not bound to a user), then a set of workflow instances are picked up in parallel and some may be processed out-of-order. It is important to note that increasing the number of **threads** on a **single** process server results in higher throughput than splitting the threads across two servers. Typically, the bottleneck of performance is the database server rather than the process servers.

If you already have a system that is deployed with multiple Workflow Agents, consider either: (1) **Stopping** the secondary agents, and increasing the threads on the primary Workflow Agent server to be the sum of the threads across the other servers, or (2) **Restricting** the secondary agents to be exclusive for the set of power users.

7.6 Graphics Section Performance

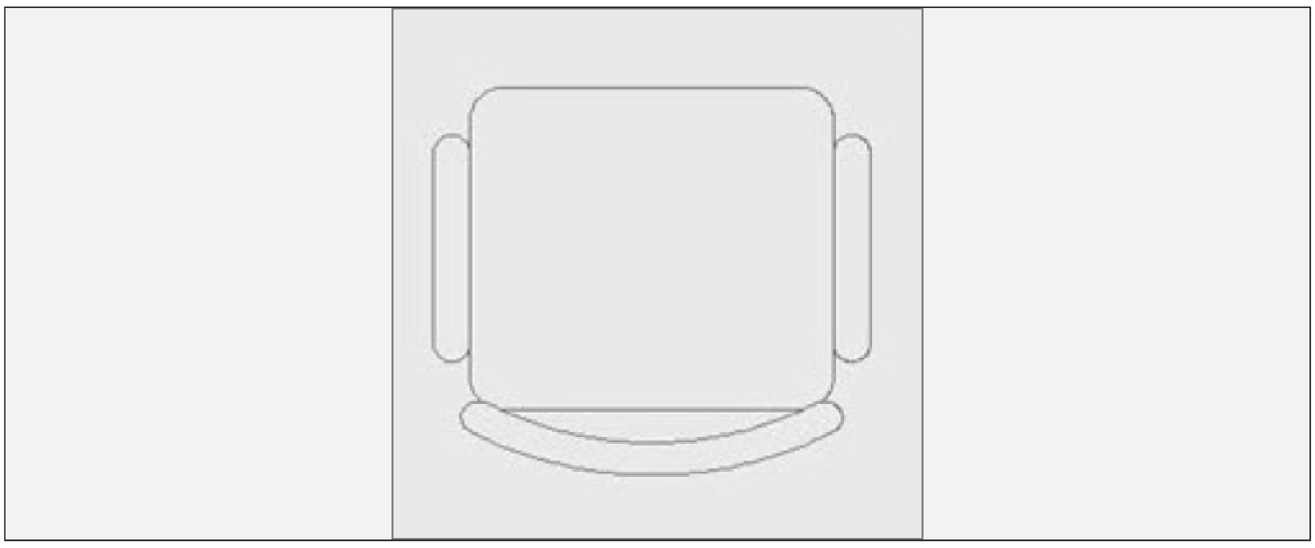
7.6.1 Drawing Size

There is often a misconception that the size of a drawing, whether it is the physical area that it represents in the world or the number of bytes it occupies in a computer drive, has a direct impact on how long it takes to display in a TRIRIGA graphics section. However, the main factor affecting performance is the number of **entities** that are visible at the time that the graphic is rendered to the user. Often, the relative or perceived drawing "size", physical or digital, has little correlation to these entities.

An **entity** is an individual object or element, such as a line segment, that exists in a CAD drawing. (In MicroStation, these are generally referred to as "elements".) A **polyline**, which can be comprised of multiple line and arc segments, is also an entity, and it is a single entity regardless of how many vertices and segments it contains. In many cases, CAD drawings contain layers with information about walls, doors, furniture, and so on, that are comprised of "exploded" singular entities rather than polylines.

For example, the following chair might look simple. Visually, it is only 4 distinct shapes and can be easily represented by 4 polylines. However, in this case, it is actually comprised of 27 individual entities, consisting of disconnected arcs and line segments. The difference between 4 and 27 is not large, but imagine a furniture layer on a floor plan that contains 500 chairs, which is not unusual or excessive. A layer that can be comprised of only 2000 entities has 13,500 instead, which increases the load time for that single layer by a factor of nearly 7x. Normally, any drawing that contains furniture blocks in this state can have the same issue with other components of the drawing, which can have a major overall impact on performance.

Example of Chair: 4 Polylines vs. 27 Entities.

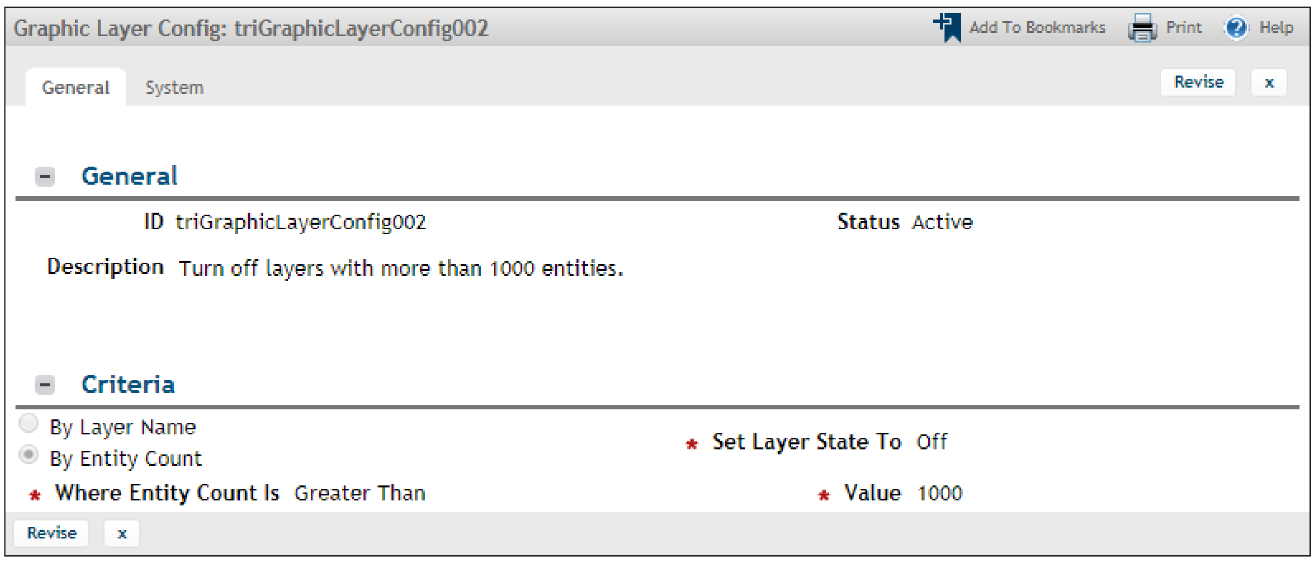


7.6.2 Graphic Layer Config

One way to improve performance on the TRIRIGA side is to limit what is initially loaded for all users in the system. This can be accomplished via **Graphic Layer Config** records, which are normally found under **Tools > Administration > Graphics > Layer Configuration**. By default, we include a record that initially turns all layers **off** with an entity count greater than 1000.

This number can be much **lower** to have a more dramatic effect. It can even be lowered to 1. It is important to note that Graphic Layer Config records have **no** impact on attached layers, so even if you set this value to 1, all of your attached polyline layers will still load completely. This is ideal for most use cases, and as a result, graphics across the system will load quickly regardless of their perceived "size". Users that require more information at the time of display can use the **Layer Manager** to turn on the layers they need to view or export.

Graphic Layer Config: Turn Off Layers With More Than 1000 Entities.



Graphic Layer Config: triGraphicLayerConfig002

General

ID triGraphicLayerConfig002

Status Active

Description Turn off layers with more than 1000 entities.

Criteria

○ By Layer Name
● By Entity Count

* Set Layer State To Off

* Where Entity Count Is Greater Than

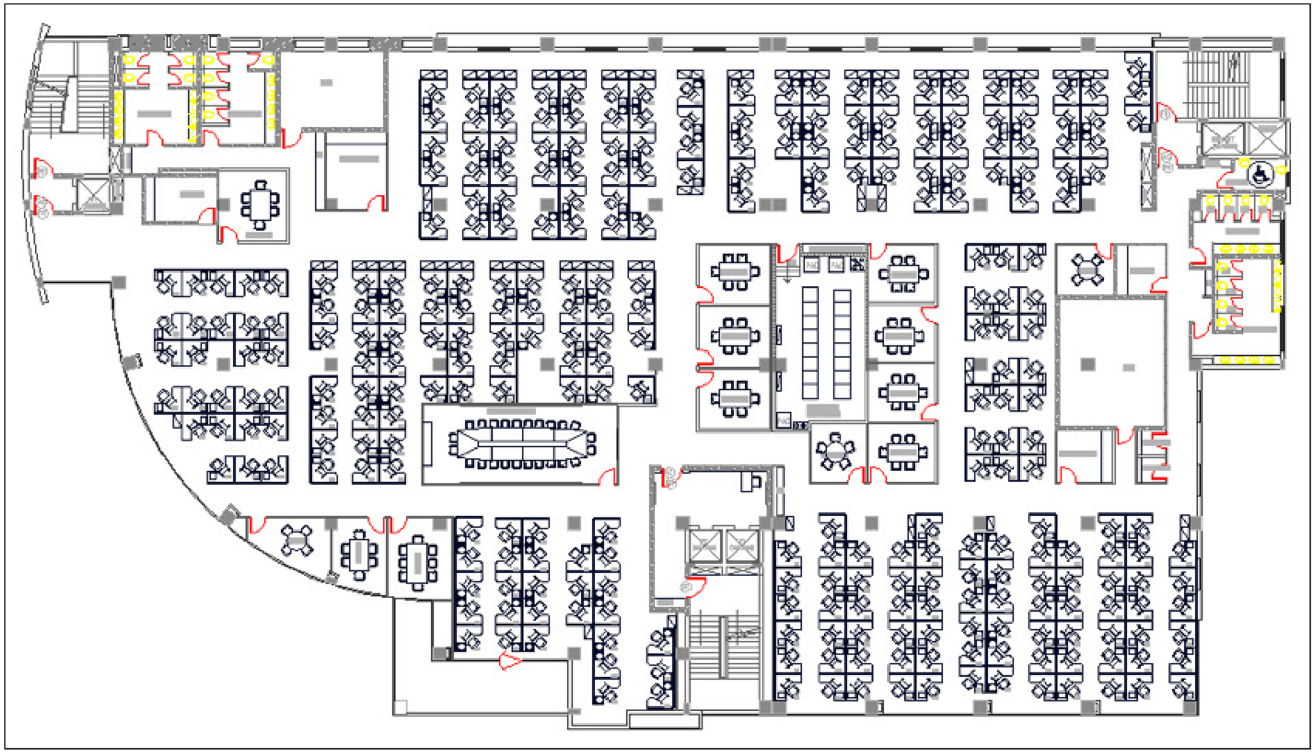
* Value 1000

7.6.3 Simplifying Drawings

Next, if the more-complex layers that a Graphic Layer Config normally turns off are used fairly **often**, the ideal way to limit load times is to go back through your CAD drawings and clean them up by joining exploded blocks and other unconnected entities. While this might seem like a large effort, the process can actually be very easy and can dramatically improve performance. The best way to demonstrate this process is with a real-world example, such as an AutoCAD floor plan.

The following AutoCAD architectural reference file contains an excessive amount of geometry, especially on the furniture layer. When published "as-is" to a graphics section, the drawing renders all of the layers in about 50 seconds, where 45+ seconds of that is the furniture layer. In order to improve the time, we will start by reducing the number of **entities** in the furniture blocks.

Example of AutoCAD Floor Plan.



a. CAD Commands

Notes:

- If there are repeated furniture in your drawing that are **not** in blocks or cells, the first step is to create **blocks** or cells so you are not editing the same furniture many times.
- This example assumes that blocks are **already** created for the furniture.

In our example, the following desk-and-chair workstation is represented by a **block** and is repeated in each cubicle 350 times. To display this simple workstation, there are 101 individual entities for this block, or over 35,000 entities on the furniture layer. In AutoCAD, one quick way to simplify this block is to take advantage of the **JOIN** option of the **PEDIT** command. After initiating the **PEDIT** command, you can select all of the entities, convert them to polylines if applicable, and join them at a given threshold.

In less than 10 seconds, this **JOIN** command reduced the total number of entities for this block from 101 to 29, and reduced the total number of entities on the furniture layer by over 25,000. More importantly, because this can all be done via the command line, it can be easily **automated** via a simple LISP routine. If you are unfamiliar with writing LISP that can open a set of drawings in batch, there are existing free utilities like StarBatch that can do this work for you. Beyond this step, you just need your script to call the noted commands on the requisite layers. For more information, see AutoCAD Command Line API Specification.

b. Manual Editing and Cleanup

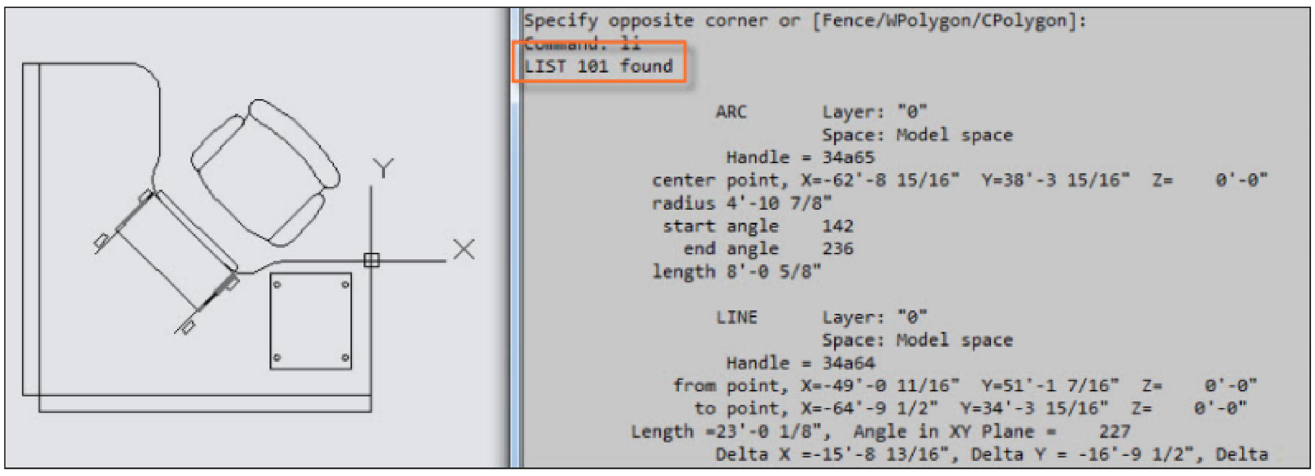
As a more manual approach for trouble areas and even more dramatic improvement, you have the option to remove unneeded geometry or redraw some of it by hand. Although this approach might take 10 minutes instead of 10 seconds, the desk and chair in their simplest forms can be represented with 2 polylines alone.

7.6.4 Summary

When applying the use of the **PEDIT JOIN** option, as well as some manual simplification across the entire floor plan, the load time in the graphics section with all layers turned **on** is reduced from 50 seconds to less than 5 seconds.

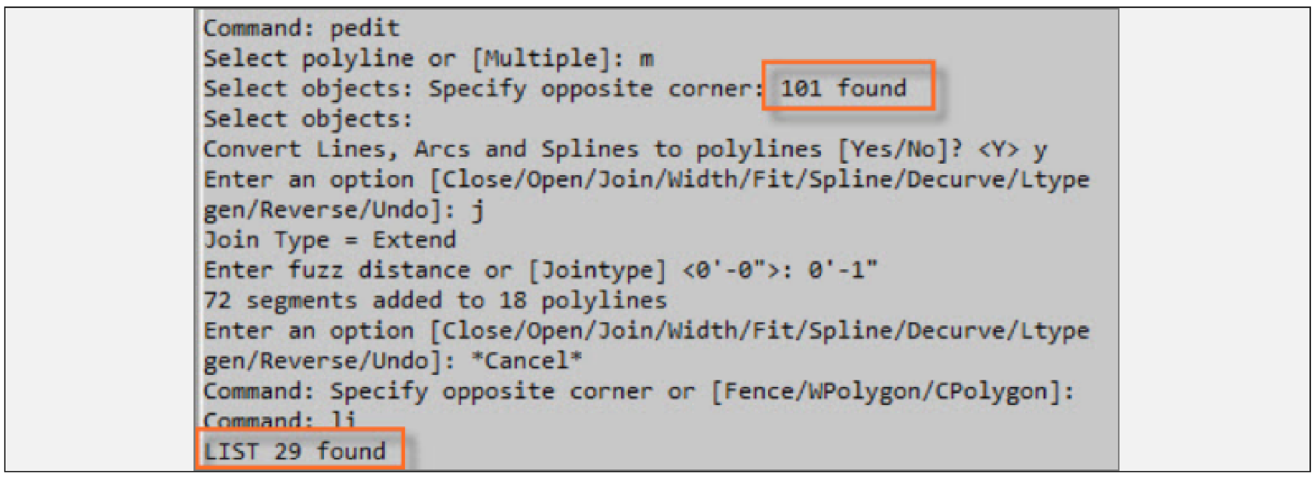
In summary, for the initial display of a graphic, it is often best for the typical use case to load only what you need with Graphic Layer Config records. In most cases, users will use the graphics section for reporting, interacting, or locating attached entities like spaces. In these scenarios, extraneous detail often only serves to clutter the graphic. For users who want the option to display more-complex layers with furniture or architectural detail, the load times across the system can be dramatically improved by simplifying your drawings through the built-in features of your CAD application, or through manual editing and cleanup.

Example of Desk & Chair Workstation: 101 Entities.



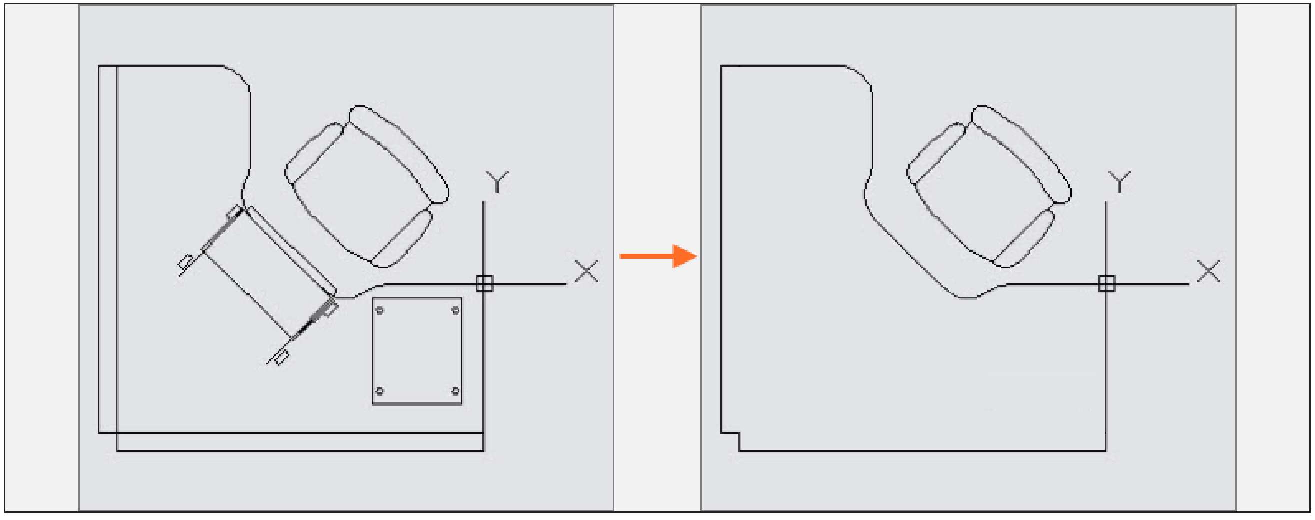
```
Specify opposite corner or [Fence/WPolygon/CPolygon]:
Command: li
LIST 101 found

                  ARC       Layer: "0"
                            Space: Model space
                   Handle = 34a65
            center point, X=-62'-8 15/16"  Y=38'-3 15/16"  Z=    0'-0"
            radius 4'-10 7/8"
             start angle    142
               end angle    236
            length 8'-0 5/8"

                  LINE      Layer: "0"
                            Space: Model space
                   Handle = 34a64
              from point, X=-49'-0 11/16"  Y=51'-1 7/16"  Z=    0'-0"
                to point, X=-64'-9 1/2"  Y=34'-3 15/16"  Z=    0'-0"
          Length =23'-0 1/8",  Angle in XY Plane =    227
                  Delta X =-15'-8 13/16", Delta Y = -16'-9 1/2", Delta
```

Example of AutoCAD Command Sequence: 29 Entities.



```
Command: pedit
Select polyline or [Multiple]: m
Select objects: Specify opposite corner: 101 found
Select objects:
Convert Lines, Arcs and Splines to polylines [Yes/No]? <Y> y
Enter an option [Close/Open/Join/Width/Fit/Spline/Decurve/Ltype
gen/Reverse/Undo]: j
Join Type = Extend
Enter fuzz distance or [Jointype] <0'-0">: 0'-1"
72 segments added to 18 polylines
Enter an option [Close/Open/Join/Width/Fit/Spline/Decurve/Ltype
gen/Reverse/Undo]: *Cancel*
Command: Specify opposite corner or [Fence/WPolygon/CPolygon]:
Command: li
LIST 29 found
```

Example of Desk & Chair Workstation: Manual Editing & Cleanup.



Next >

Comments (0) | Versions (43) | Attachments (8) | About

There are no comments.

Add a comment

Feed for this page | Feed for these comments