IBM Control Center Performance and Tuning Guide

How to Find and Trouble Shoot Performance Issues, Tuning Tips, and More

Revision 0.965

March 27, 2020

Table of Contents

Revision History

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Dale Ander	Initial draft	Jan 14, 2019
0.2	Dale Ander	Refined and added info on OSA servers	Jan 15, 2019
0.3	Dale Ander	Refined and added info on B2Bi servers and other sections	Jan 16, 2019
0.4	Dale Ander	Refined, rearranged and added information on monitoring servers, including B2Bi servers and other sections	Jan 17, 2019
0.5	Dale Ander	Incorporated feedback from reviewers.	Jan 18, 2019
0.6	Dale Ander	Added information in Reporting and non-OSA server sections.	Jan 22, 2019
0.7	Dale Ander	Incorporated feedback from draft copy readers.	Jan 22, 2019
0.8	Dale Ander	Incorporated feedback from draft copy readers.	Jan 22, 2019
0.9	Dale Ander	Minor updates, including one to the definition of `BYPASS_EVENT_MONITOR_FOR_EVENTS`.	Feb 5, 2019
0.91	Dale Ander	Updated for ICC 6.1.2.1 Tables added with Engine properties and configuration settings that affect performance.	Aug 27, 2019
0.92	Dale Ander	Incorporated feedback on 0.91.	Aug 27, 2019
0.93	Dale Ander	Added link to ICC Documentation on ICC Database tuning.	Sep 25, 2019
0.94	Dale Ander	Added more details on how to turn on logging in the Web server to facilitate seeing queries initiated by the Web console and the time they take to run.	Nov 4, 2019
0.95	Dale Ander	Updates to list of properties that do not require a restart.	Nov 26, 2019
0.96	Dale Ander	Added documentation on the engine property `SUMMARY_TABLES_PURGE_FILE_COUNTS`	Feb 14, 2020
0.965	Dale Ander	Updated default for `EVENT_MONITOR_EVENT_LIMIT_PER_RUN` to reflect the current value.	Mar 27, 2020

1 Introduction

Performance issues in IBM Control Center (ICC) manifest themselves in multiple ways including:

- Slow web console performance
- SLCs causing false alerts
- Rules that send emails doing so late
- Learning about events on monitored servers, including Sterling File Gateway (SFG), late
- Getting events from Open Server Architecture (OSA) type servers, including IBM Global Mailbox, late
- Information about completed processes and/or file transfers appearing late in the Web console

Many, but not all, of these issues are related to slow database performance, and that's typically where you should look first.[1]

Besides the database, SLC events are impacted by the performance of the EventMonitor service in multi-Event Processor (EP) environments.

OSA server performance is impacted by the performance of the EventProcessorService logic and an ICC servlet that inserts data received from OSA servers into the ICC `CC_UNPROCESSED_EVENT` database table.

Data shown in the ICC Web Console Completed Process and Completed File Transfer views, as well as the Recent file transfer activity dashboard widget, is controlled by the Process Summary logic, which is impacted by database performance.

And all these things could be impacted by erroneous rules and actions!

This document is intended to help you the reader, whether you are an administrator of ICC, or an ICC support person, identify why ICC performance issues are occurring as well as provide tips on both avoiding and resolving them.

[1] For documentation on tuning ICC's database see https://www.ibm.com/support/knowledgecenter/SS4Q96_6.1.0/com.ibm.help.scc.tuning.doc/SCC_Tune_database_server.html

2 Database

ICC, especially as of 6.1, relies heavily on its database to provide functionality. If the database queries – inserts, updates, and deletes, initiated by ICC run slowly, so will ICC. To know if the database is responsible for performance problems you should first do a few text-based searches in the ICC engine logs. You can start by looking for the following strings:

- `CCTR135I`
- `CCTR138I`
- `excessive`
- `inordinate`

Messages logged with this text may look like this:

- `CCTR135I Time to insert monitored server data into EVENTS database table was: 436776 milliseconds`
- `CCTR138I Time EventMonitor took to retrieve data from EVENTS database table was: 1254366 milliseconds`
- `updateQueuedProcesses query time excessive`
- `updateQueuedProcesses query time (t1-t0) excessive - Total time`
- `ProcessSummaryWorker.getEvents() logic took an inordinate amount of time!`
- `Time to update nnn processes as ended was excessive.`
- `ProcessSummaryWorker took an inordinate amount of time to summarize file transfer`
- `ProcessSummaryWorker took an inordinate amount of time to summarize Process`
- `ProcessSummaryWorker took an inordinate amount of time to commit database changes to CC_PROCESS and CC_FILE_TRANSFER`

These messages are indicative of:

- Databases whose indices are not being maintained on a regular basis and/or
- Databases that have too much data in them for customers to get the performance they require.

Database inserts and updates should typically take just a few milliseconds - they should not take minutes. The first thing you might do when you find queries are taking too long is ensure the ICC table indices are being maintained on a regular basis.

TIP – You may write rules to watch for events whose message IDs are `CCTR135I` or `CCTR138I` to avoid having to look at the ICC engine log files to know when these messages are being logged, which would allow you to be proactive on performance issues.

Every 6 hours, another set of metrics are output that contains valuable database related metrics (and more) for that time frame. One that looks like this:

```
11 Jan 2019 22:33:36,577 280851610 [LicenseChecker] INFO  SCCAgentLicenseCheck -
--------Java Heap Memory Details------------
Max Heap Size = 4194304 KB
Current Max Heap Size = 2090560 KB
Free  Memory = 2334493 KB
Used  Memory = 1859810 KB
-----------------------------------------
11 Jan 2019 22:33:36,577 280851610 [LicenseChecker] INFO  EventProcessing -
<PRE>---------------------- Event processing Timings ----------------------

TOTAL_EVENTS_PROCESSED ...................: 5762716
TOTAL_TIME_FOR_GETTING_RULE_SESSION .....: 2065
TOTAL_TIME_FOR_RULE_MATCH ................: 155706
TOTAL_TIME_FOR_RULE_QUERY ................: 5629
TOTAL_TIME_FOR_ACTION_PROCESSING .........: 814011
TOTAL_TIME_FOR_ATTRIBUTE_SETTING .........: 235
TOTAL_TIME_FOR_RE_PROCESSING .............: 978645
Average Time for Rule Processing .........: 0.169824
Average Time for Rule Match ..............: 0.027020

-----------Monitoring monitoring-------------
Average Time to check on polling status .: 0.036744

-----------Action Processing Details----------------------
TOTAL_TIME_FOR_ACTION_PROCESSING .........: 814011
TOTAL_TIME_FOR_BROADCASTING ..............: 4089

-----------Database Access Timings for CD_STATS_LOG Table ----------------------
TOTAL_CDSTATS_INSERTED             .........: 14968
TOTAL_CDSTAT_INSERTS               .........: 3929
TOTAL_TIME_FOR_CDSTAT_INSERTS ..............: 10828
Average Time for CD_STATS_LOG table Inserts: 2.755918
Average Time for CD_STATS_LOG Insert .......: 0.723410

-----------Database Access Timings for EVENTS Table ----------------------
TOTAL_EVENTS_INSERTED              .........: 212042
TOTAL_EVENT_INSERTS                .........: 128874
TOTAL_TIME_FOR_EVENT_INSERTS ..............: 605630
Average Time for EVENTS table Inserts  ...: 4.699396
Average Time for EVENT insert          ...: 2.856179

TOTAL_EVENT_UPDATES                .........: 4244
TOTAL_TIME_FOR_EVENT_UPDATES ..............: 58135
Average Time for EVENTS table Updates  ...: 13.698162
</PRE
```

TIP – Performance of ICC may be impacted by the amount of heap memory allocated to it. When too little heap memory is allocated to ICC – something done via the command used to start ICC, but just enough to avoid Out-of-memory exceptions, it can cause the Java garbage collection logic to run more often, which has a negative impact on performance of ICC. The "Java Heap Memory Details" logged help you ascertain if more memory for the Java heap could help ICC performance. It is advised you never allocate less than 4GB for heap.

Primarily you will check the data above to see how well the database performed for the previous 6 hours (the values shown are all reset after they're logged). In this output, times for inserts and updates for the `EVENTS`, and `CD_STATS_LOG`, tables are in milliseconds and should not be too large. What's too large? In general, double digit values for these numbers are too large. Database inserts and updates should certainly not take minutes or even 10s of seconds.

You may wonder what the difference is between `TOTAL_CDSTATS_INSERTED` and `TOTAL_CDSTAT_INSERTS`, and `TOTAL_EVENTS_INSERTED` and `TOTAL_EVENT_INSERTS` in the

metrics data above. ICC typically inserts multiple rows at a time, i.e. ICC strives to do batch inserts, instead of inserting one row at a time, because doing so is more efficient. `TOTAL_CDSTATS_INSERTED` and `TOTAL_EVENTS_INSERTED` are the number of rows inserted for the previous 6 hours, while `TOTAL_CDSTAT_INSERTS` and `TOTAL_EVENT_INSERTS` are the number of inserts performed to insert those rows.

TIP – Databases have performance gathering and reporting tools that may be used to help ferret out performance issues. E.g. Oracle has Automatic Workload Repository (AWR). The database administrator for the system should be able to use these tool(s) to help find, recommend changes, and resolve many problems causing performance issues.

2.1 Database Indexes

Customers need to ensure the ICC database table indices are maintained on a regular basis. If customers do not have a process in place to do so, they will need to. Indices that are critical to be maintained for good performance include:

Table	Index	Comment
EVENTS	SEQ_NUM_INDEX	Used to prevent duplicate inserts
EVENTS	EVENT_ID_INDEX	
EVENTS	EVENTS_PKKEY_IX	
EVENTS	EVENTS_STAT_INDEX	Used to find data by the process summarization logic
EVENTS	EVENTS_STAT_IDX	Used by a thread named QueuedProcessesClearJob, which runs frequently within ICC
MQ_STATS_LOG	MQ_EVENT_ID_IX	
MQ_STATS_LOG	MQ_STAT_INDEX	
MQ_STATS_LOG	MQ_PKEY_IX	
AF_STATS_LOG	AF_EVENT_ID_IX	
AF_STATS_LOG	AF_STAT_INDEX	
AF_STATS_LOG	AF_PKEY_IX	
BP_STATS_LOG	BP_EVENT_ID_IX	
BP_STATS_LOG	BP_STAT_INDEX	
BP_STATS_LOG	BP_PKEY_IX	
FG_STATS_LOG	FG_EVENT_ID_IX	
FG_STATS_LOG	FG_STAT_INDEX	
FG_STATS_LOG	FG_PKEY_IX	
EVENTS_EXT	EVENTS_EXT_INDEX	Used by summarization logic
EVENTS_EXT	EVENTS_EXT_PKEY_IX	
CC_PROCESS	CC_PROCESS_IDX1	
CC_PROCESS	CC_PROCESS_IDX2	
CC_PROCESS	CC_PROCESS_IDX3	
CC_PROCESS	CC_PROCESS_IDX4	
CC_PROCESS_DVG		Only has a primary key index
CC_FILE_TRANSFER	CC_FILE_TRANSFER_IDX1	

Table	Index	Comment
CC_FILE_TRANSFER	CC_FILE_TRANSFER_IDX2	
CC_FILE_TRANSFER	CC_FILE_TRANSFER_IDX3	
CC_FILE_TRANSFER	CC_FILE_TRANSFER_IDX4	
CC_FILE_TRANSFER	CC_FILE_TRANSFER_IDX5	
CC_FILE_TRANSFER_DVG		Only has a primary key index
ROLL_UP	RU_INDEX	

TIP – Partitioning the ICC database tables improves performance of ICC queries, including those used to purge data. If you have not partitioned the ICC database tables – an option presented when the ICC `configCC` script it run after ICC is first installed, you should.

You can know if the ICC database tables have been partitioned by looking for the message logged when ICC runs its purging logic, also known as "the daily operation". E.g.

```
14 Jan 2019 16:49:54,408 24004 [CCEngine(CCDale1)] INFO  JDBCService - Database is partitioned and doing daily operation...
```

You can also look in the BulkDataMover log file (and this is probably the best place to look) to see if the ICC database tables have been partitioned. E.g.

```
14 Jan 2019 16:49:47,095 16691 [CCEngine(CCDale1)] INFO  BulkDataMover - All tables have been partitioned and determined that partitioning has been done...
```

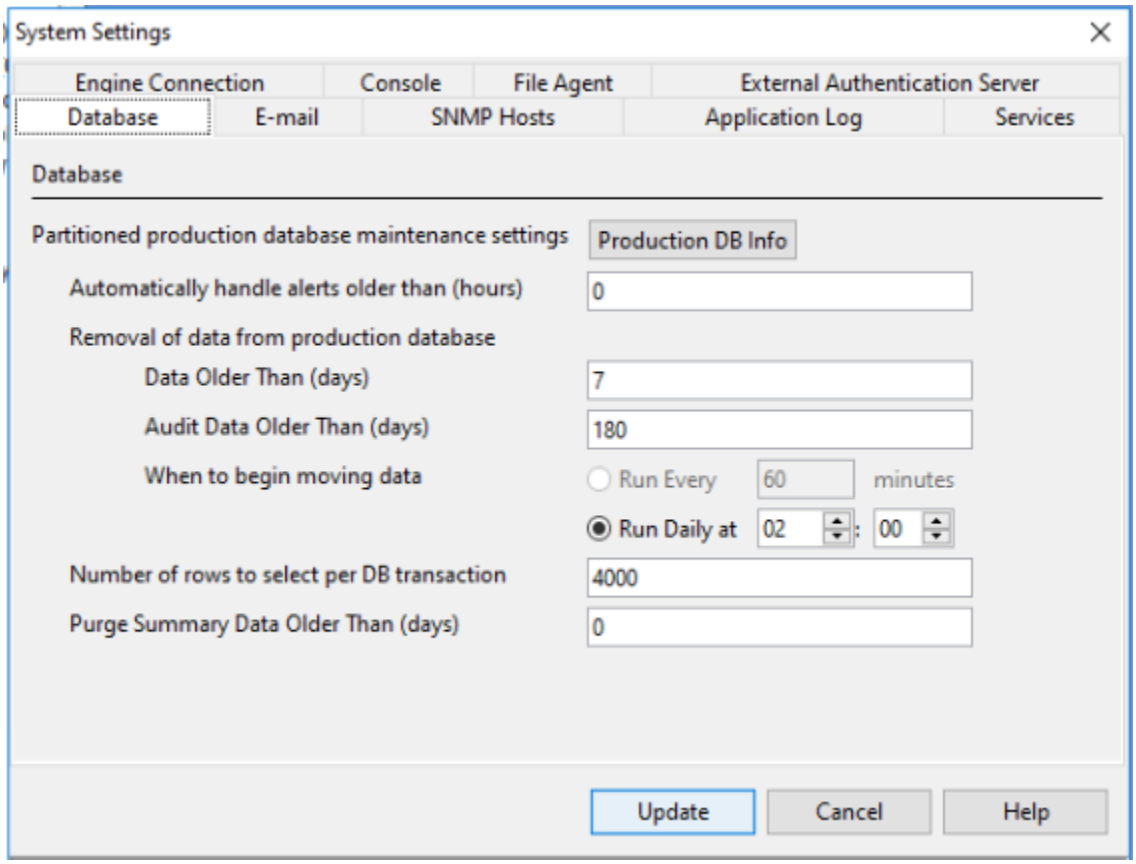
2.2 *Database Data*

Poor database performance may be incurred simply because you are retaining too much data, relative to your database server's "ability", in the ICC tables to get the performance you require. This is especially true of the ICC Summary tables as they are currently (as of January 2019) not able to be partitioned (unlike other ICC database tables).

The ICC Summary table names are:

- `ROLL_UP`
- `CC_PROCESS`
- `CC_PROCESS_DVG`
- `CC_FILE_TRANSFER`
- `CC_FILE_TRANSFER_DVG`

Newer releases of ICC, 6.1.1.0 and later, allow you to configure ICC to automatically truncate data from them after a specified number of days via the Swing console:



TIP – By default, the "Purge Summary Data Older Than (days)" setting (seen in the Swing console dialog above) is zero, which tells ICC to **never** purge data from the summary tables. **You should change this, but before you do, you should follow the directions to Manually Purging Summary Table Data.** "Purge

Summary Data Older Than (days)" should be at least as large as “Data Older Than (days)”, but maybe not larger.

When ICC is configured to purge summary table data two engine properties are utilized to do so:

- SUMMARY_TABLES_PURGE_BATCH_SIZE and
- SUMMARY_TABLES_PURGE_QUEUE_SIZE

SUMMARY_TABLES_PURGE_QUEUE_SIZE, whose default is 100000, sets the limit on the SQL used to ascertain the data to be deleted from the summary tables.

SUMMARY_TABLES_PURGE_BATCH_SIZE, whose default is 100, is used as a limit in the SQL used to delete summary table data. Since the SQL used to delete data causes far more contention on the database than the SQL used to find the data to be deleted, the value for this property should be kept relatively small.

Note just setting “Purge Summary Data Older Than (days)” to a value greater than zero will not cause all summary data to be purged. Another engine property, SUMMARY_TABLES_PURGE_ROLL_UP, whose default is FALSE, tells ICC whether to purge data from ROLL_UP when other summary data is purged. (ROLL_UP table data is used by the ICC Web console to facilitate the Recent file transfer activity and Transfer scorecard dashboard widgets.) By setting SUMMARY_TABLES_PURGE_ROLL_UP to TRUE, data from ROLL_UP will be purged at the same time data from the other summary tables is purged.

Database performance is also affected by the amount of data retained in the EVENTS table, and its ancillary tables, which include all the _STATS_LOG tables – BP_STATS_LOG, CD_STATS_LOG, etc. The amount of data retained in these tables may be controlled via the value specified for “Data Older Than (days)” in the same dialog shown above.

TIP – Data purging should **not** be scheduled to run during a busy processing time by ICC. I.e. Data purging should not be scheduled when the servers monitored by ICC are at their busiest. The time at which the daily removal of data should be performed should coincide with the time at which ICC is generating the fewest number of events.

The following Microsoft SQL server query (it will need to be adjusted for other database types) returns the number of events generated per hour and can be used to ascertain a good time to schedule the daily data purge:

```
SELECT convert(char(14), DATE_TIME,126) as dtm, count(EVENT_ID) as number  FROM EVENTS (nolock)  group by convert(CHAR(14),DATE_TIME,126)  order by dtm
```

*2.2.1 **Manually Purging Summary Table Data***

If the version of ICC you're running is old (prior to ICC 6.1.1) and does not have the ability to automatically purge the summary table data, this section provides instructions on how to do so manually.

TIP – Even if your version of ICC does support purging of the summary table data, if you've allowed more summary table data to accumulate than you now want to keep, it's wise to perform a manual purge, once, before turning on the ability in ICC to do so.

The five database tables with summary data are:

- `ROLL_UP`
- `CC_PROCESS`
- `CC_PROCESS_DVG`
- `CC_FILE_TRANSFER`
- `CC_FILE_TRANSFER_DVG`

Note if you're not using DVGs (data visibility groups) in ICC, no data will be written to `CC_PROCESS_DVG` or `CC_FILE_TRANSFER_DVG`, which leaves only three tables for you to maintain but we'll cover what you need to do though as if you had DVGs...

First pick a date that represents the oldest summarization data you want to keep, for example May 19th, 2017

`ROLL_UP` is straight forward to "truncate" because `DATE_TIME`, the column you need to reference in SQL, is just a "varchar" kind of column (as opposed to a "date" type, so SQL to truncate it would just be, for example:

```
DELETE FROM ROLL_UP WHERE DATE_TIME < '2017-05-20'
```

The other SQL you'll need to run is slightly "trickier" as you need to convert a string to a date in either Oracle, DB2, or MSSQL SQL, which I must leave to you... But other than that, the SQL is very straight forward

Perform this SQL in this order

- `DELETE FROM CC_PROCESS_DVG WHERE PROCESS_ID IN (SELECT PROCESS_ID FROM CC_PROCESS WHERE STARTED < your date)`
- `DELETE FROM CC_PROCESS WHERE STARTED < your date`
- `DELETE FROM CC_FILE_TRANSFER_DVG WHERE FILE_TRANSFER_ID IN (SELECT FILE_TRANSFER_ID FROM CC_FILE_TRANSFER WHERE STARTED < your date)`
- `DELETE FROM CC_FILE_TRANSFER WHERE STARTED < your date`

3 EP Startup and Shutdown

When EPs start, they start a NodeService thread for each monitored server (there is a one-to-one correspondence between NodeService threads and monitored servers – not counting OSA servers). Prior to ICC 6.1.2.1, NodeService threads were started, and stopped, serially (one at a time) at EP startup and shutdown. If you're running ICC 6.1.2.1, or later, EPs are configured, by default, to start and stop multiple NodeService threads in parallel at EP startup and shutdown. This behavior is controlled by two engine properties:

- `USE_MULTIPLE_THREADS_FOR_NODESERVICE_STARTUP_AND_SHUTDOWN`
- `SERVER_STARTUP_SHUTDOWN_THREADS`

The default value for `USE_MULTIPLE_THREADS_FOR_NODESERVICE_STARTUP_AND_SHUTDOWN` is true. And the default value for `SERVER_STARTUP_SHUTDOWN_THREADS` is the number of CPUs ICC ascertains the server ICC is run on has. Unless you configure the system otherwise, NodeService threads will be started and stopped in parallel as of ICC 6.1.2.1.

For the fastest EP startup and shutdown times, it is not advisable to change these default values.

4 Server Reassignment

Prior to ICC 6.1.2.1, when monitored servers were reassigned, typically because the CEP ascertained a previously running EP has stopped, or a server rebalance was initiated, they were reassigned one at a time, serially. Each NodeService thread was stopped on its previous active EP, if the EP was still running, and then restarted on its new active EP. If you're running ICC 6.1.2.1, or later, server reassignments will be done, by default, in parallel, which speeds up the reassignment process significantly. This is controlled by two engine properties:

- `MULTI_THREAD_SERVER_REASSIGNMENT`
- `SERVER_REASSIGNMENT_QUEUE_SIZE`
- `SERVER_REASSIGNMENT_THREADS`

The default value for `MULTI_THREAD_SERVER_REASSIGNMENT` is true. The default value for `SERVER_REASSIGNMENT_QUEUE_SIZE` is 100000. And the default value for `SERVER_REASSIGNMENT_THREADS` is the number of CPUs ICC ascertains the server ICC is run on has.

For the fastest server reassignment processing, it is not advisable to change these default values.

5 EventMonitor

The ICC EventMonitor service only ever runs in the Controller Event Processor (CEP) and is responsible for passing all events created by all Event Processors (EPs) to the SLC (and FileAgent) service.

TIP – When running ICC with a single EP, you should always specify the engine property `BYPASS_EVENT_MONITOR_FOR_EVENTS` with a value of `TRUE`. When `BYPASS_EVENT_MONITOR_FOR_EVENTS` is `TRUE`, events will go directly to the SLC, and FileAgent, service instead of indirectly via the EventMonitor service, and contention on the ICC `EVENTS` database table will be decreased.

To know how the EventMonitor service is performing you may look in the engine log for the metrics it outputs once an hour. They could look like this:

```
24 Sep 2018 01:00:03,605 4647011 [EventMonitor] INFO  EventMonitor -                None: ******
EventMonitorMetrics (EVENT_MONITOR_EVENT_LIMIT_PER_RUN = 125, EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP =
100) ******
Current hour (Australia/Sydney): 0
Notifications broadcast this hour: 3386
Times processEvents() invoked this hour: 719
Times processEvents() invoked this hour without waiting: 0
Times processEvents() invoked this hour without waiting percent: 0
Average time to run processEvents(): 9 ms
Average time to retrieve event data: 2 ms
Average time to populate CachedResultSet: 0 ms
Average time to populate EventCacheManager: 0 ms
Average time to construct notifications: 5 ms
Average time to broadcast notifications: 0 ms
Average time to update LAST_SERIAL_NUM_PROCESSED: 0 ms
Average time to close result set, etc.: 0 ms
processEvents() retry count: 0
```

Or like this (this is the latest output style):

```
07 Jan 2019 11:00:04,290 1637440 [EventMonitor] INFO  EventMonitor -                None: ****** EventMonitor
Metrics (EVENT_MONITOR_EVENT_LIMIT_PER_RUN = 125, EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP = 100)  Hour
(America/Chicago): 10 ******
1  EventMonitor processEvents() invoked = 2875
1b EventMonitor processEvents() invoked without waiting (in catchup) = 2599
1c EventMonitor convertXMLStringsToNotificationsAndBroadcastThem() invoked = 5314
2  Events processed by processEvents() = 330655, Max value = 125, Average = 115
3  Notifications constructed and broadcast by convertXMLStringsToNotificationsAndBroadcastThem() = 330655, Max value
= 100, Average = 62
4  Milliseconds performing EventMonitor processEvents() = 231718, Max value = 1552, Average = 80
4a Milliseconds getting DB connection = 145, Max value = 31, Average = 0
4b Milliseconds running SQL to retrieve events = 533, Max value = 141, Average = 0
4c Milliseconds populating CachedRowSet = 3608, Max value = 719, Average = 1
4d Milliseconds populating EventCacheManager = 82, Max value = 16, Average = 0
4e Milliseconds updating CC_CONTROLLER.LAST_EVENT_SERIAL_NUM_PROC = 3703, Max value = 453, Average = 1
4f Milliseconds closing cached result set and database connection = 159, Max value = 64, Average = 0
4g Milliseconds closing query result set = 571, Max value = 156, Average = 0
4h Milliseconds retrieving data from CRS used to constructing XMLStrings = 318, Max value = 47, Average = 0
4i Milliseconds constructing XMLStrings = 54214, Max value = 1219, Average = 0
4j Milliseconds creating SCCNotifications and broadcasting them = 168351, Max value = 597, Average = 31
5  Milliseconds in convertXMLStringsToNotificationsAndBroadcastThem() = 168351, Max value = 597, Average = 31
5a Milliseconds broadcasting SCCNotifications = 943, Max value = 125, Average = 0
5b constructing SCCNotifications = 167408, Max value = 596, Average = 31
6  Milliseconds constructing XMLStrings, converting them to notifications and broadcasting them = 222998, Max value
= 1525, Average = 77
7  Milliseconds delta between ACTION_COMPLETED and reconstituting events = 85501, Max value = 4953, Average = 2590
```

One key metric to look for in the data logged is the number of times `processEvents` was invoked relative to how many times it was invoked “without waiting”. When `processEvents` is invoked without waiting, it is an indication that the EventMonitor service is unable to keep up with the current load of events. And this may, or may not, be related to poor database performance (but it typically is).

Note when the times `processEvents` is invoked matches the times `processEvents` is invoked without waiting it means the EventMonitor service was not keeping up with the number of events generated by the system at any point during the hour prior to the metrics being logged.

TIP – If SLCs are incorrectly generating events about things not happening, you can use these two queries to see if the EventMonitor is currently behind in getting events to the SLC service, and if so, by how far:

- `SELECT LAST_EVENT_SERIAL_NUM_PROC FROM CC_CONTROLLER`
- `SELECT MAX (SERIAL_NUM) FROM EVENTS`

The first query's result tells the "serial number" value of the event the EventMonitor last processed – it's a unique value assigned by the database as data is inserted into the `EVENTS` table, while the result of the second query tells the "serial number" of the last event in `EVENTS` that needs to be processed. The difference between those two values is approximately how far "behind" the EventMonitor logic is. The values should typically be close (within a thousand or two).

In ICC 6.1.2 releases after August 2019 the metric to show the average delta between `ACTION_COMPLETED` and when an event was reconstituted was added. E.g.

```
7  Milliseconds delta between ACTION_COMPLETED and reconstituting events = 85501, Max value = 4953, Average = 2590
```

Since the `ACTION_COMPLETED` value represents the time, in milliseconds since January 1, 1970 an EP completed processing of an event, this metric gives a user an approximation of the delay between when an event was processed by one EP and its delivery to other services, including the SLCService, by the CEP. When this delay is long, and SLC schedule tolerances are small, erroneous SLC events are likely to occur. You want this maximum value and average to be as small as possible.

Other important metric values to look at include:

- `Average time to retrieve event data`
- `Average time to construct notifications`
- `Average time to broadcast notifications`
- `Milliseconds running SQL to retrieve events = 533, Max value = 141, Average = 0`
- `Milliseconds broadcasting SCCNotifications = 943, Max value = 125, Average = 0`
- `constructing SCCNotifications = 167408, Max value = 596, Average = 31`

When the values for "`Average time to retrieve event data`" or "`Milliseconds running SQL to retrieve events`" are large – queries should take milliseconds, not 10s or minutes of seconds, database performance is having a negative impact on the performance of the EventMonitor service.

When the values for "`Average time to construct notifications`" or "`constructing SCCNotifications`" are large, CPU limitations may be having a negative impact on the performance of the EventMonitor service. Note that constructing notifications also requires database I/O. Increasing the value for the engine property `EVENT_MONITOR_THREADS`, described below, may improve this performance.

When the values for "`Average time to broadcast notifications`" or "`Milliseconds broadcasting SCCNotifications`" are large – more than a few milliseconds, it could be an indication that you are using "Alert 0" actions. Erroneous use of "Alert 0" actions can be a major source of performance problems in ICC. Refer to the section Rules and Actions

As ICC events are generated, they are processed by the RuleService in the EP they originated from, and the number of rules defined, and the actions associated with rules that are triggered, can have a large impact on the performance of ICC.

TIP – Disable or delete all ICC rules, including built-in rules, that are not being used to improve ICC performance.

ICC can typically handle having 1000s of enabled rules without incurring performance problems, but while some actions, e.g. email actions, have little impact on ICC performance, others can have a large, negative, impact, e.g. OS Commands and Alert 0 actions.

5.1 OS Commands

OS Commands specify a script, or batch file, ICC is to perform when a rule is triggered by an event. When a script or batch file initiated by ICC takes a long time to run it delays processing of other events by ICC so be sure they always run quickly.

TIP – When scripts or batch files will take more than a few milli-seconds to run, have them execute whatever they do asynchronously, instead of synchronously, when possible.

Alert 0 for more details.

5.2 EventMonitor Properties

Multiple engine property values may be changed to alter the performance of the EventMonitor service, including:

- `EVENT_MONITOR_EVENT_LIMIT_PER_RUN`
- `EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP`
- `EVENT_MONITOR_THREADS`
- `EVENT_MONITOR_JUST_SLC`[2]

`EVENT_MONITOR_EVENT_LIMIT_PER_RUN` and `EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP` dictate how much data the EventMonitor service will attempt to read from the `EVENTS` table in one batch (it's the database query limit), and the count of records retrieved at which it will enter "catchup mode", respectively. (In "catchup mode" the EventMonitor service will not wait before querying for more data to process.) The default values for these properties are 1000 and 200[3], respectively, and increasing them may or may not help with performance but you may try.

TIP – the `EVENT_MONITOR_COUNT_FOR_CATCHUP` value should always be made slightly smaller than the value for `EVENT_MONITOR_EVENT_LIMIT_PER_RUN` for best performance. This way even if the EventMonitor does not get limit events to process, if it gets a big percentage of the limit it will immediately look for more to process rather than waiting before it "looks" again.

`EVENT_MONITOR_THREADS` is a new property (introduced in all 6.1 releases of ICC in January 2019) whose default is the number of CPUs the server running the EP has[4]. In theory this property should reduce the

---

[2] Available as of ICC 6.1.2.1

[3] Prior to ICC 6.1.2.1 the default values for `EVENT_MONITOR_EVENT_LIMIT_PER_RUN` and `EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP` was 125 and 100, respectively.

[4] Prior to ICC 6.1.2.1 the default value for `EVENT_MONITOR_THREADS` was 2.

amount of time spent "constructing XMLStrings" as its value is increased. However, depending on the number of CPUs allocated to the server running the ICC CEP it may, or may not, help overall EventMonitor performance to increase its value.

EVENT_MONITOR_JUST_SLC, a new property introduced in ICC 6.1.2.1, may be set true when no Connect:Direct File Agents are monitored by ICC. Setting this property to true allows the EventMonitor logic to limit the event data retrieved, and therefore the events created and passed on, which allows it to run faster and decrease the delta between ACTION_COMPLETED and reconstituting events.

TIP – If SLCs are incorrectly generating events about things not happening, check the maximum and average milliseconds delay metric (added to ICC 6.2.1 in August 2019). If either value is large you may need to:

- Ensure ICC database table indices are being maintained
- Increase EVENT_MONITOR_LIMIT_PER_RUN and EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP
- Increase EVENT_MONITOR_THREADS
- Set EVENT_MONITOR_JUST_SLC if no Connect:Direct File Agents are monitored

6 Process Summarization

Data displayed by the ICC Web Console's Recent file transfer activity dashboard widget, and the Completed Processes, Queued Processes, and the Completed File Transfer views, comes directly from the ICC summary database tables:

- ROLL_UP
- CC_PROCESS
- CC_PROCESS_DVG
- CC_FILE_TRANSFER and
- CC_FILE_TRANSFER_DVG

The ICC ProcessSummaryService takes detailed data written to the ICC EVENTS, and other ancillary tables, and "summarizes" that information for insertion into the various ICC "summary" tables which the ICC web console queries to facilitate its displays and views.

To know how the ProcessSummaryService is performing you may look in the ICC engine log for the metrics it outputs once an hour. They could look like this:

```
****** Process Summary Metrics for the Past Hour ******
0   Events retrieved by ProcessSummaryWorker to do summarization = 13085, Max value = 431, Average = 24
0   Summarized processes for All server types = 1948
0   Summarized processes for B2B Integrator/Sterling File Gateway servers = 1942
0   Summarized processes for Sterling Connect:Direct servers = 6
0   Summarized transfers for All server types = 664
0   Summarized transfers for B2B Integrator/Sterling File Gateway servers = 64
0   Summarized transfers for Sterling Connect:Direct servers = 600
1   Milliseconds in ProcessSummaryWorker = 52468, Max value = 4633, Average = 97
1a  Milliseconds running query to get processes to be summarized = 2059, Max value = 1391, Average = 3
1b  Milliseconds performing EntityUtil.preLoad() = 1126, Max value = 520, Average = 2
1c  Milliseconds processing results of query to get processes to be summarized = 0, Max value = 0, Average = 0
1d  Milliseconds to getEvents for summarization = 37046, Max value = 4603, Average = 68
1d1 Milliseconds in getEvents to execute SQL to retrieve events for summarization = 36086, Max value = 4603, Average = 66
1d2 Milliseconds in getEvents to construct events from result set for summarization = 894, Max value = 141, Average = 1
1d3 Milliseconds in getEvents to get database connection used to retrieve events = 49, Max value = 16, Average = 0
1e  Milliseconds performing getEventsByProcess = 171, Max value = 140, Average = 0
1f  Milliseconds sorting the events prior to passing them to the summarizer = 32, Max value = 16, Average = 0
1g  Milliseconds summarizing All processes and transfers = 3956, Max value = 265, Average = 2
1g1 Milliseconds summarizing B2B Integrator/Sterling File Gateway processes and transfers = 3802, Max value = 265, Average = 1
1g1 Milliseconds summarizing Sterling Connect:Direct processes and transfers = 154, Max value = 31, Average = 25
1h  Milliseconds adding SQL to insert data into CC_FILE_TRANSFER = 238, Max value = 79, Average = 0
1i  Milliseconds committing database changes to CC_PROCESS and CC_FILE_TRANSFER = 7224, Max value = 1594, Average = 13
1j  Milliseconds to updateStatistics (Updates to ROLL_UP) = 475, Max value = 62, Average = 0
2   Milliseconds summarizing All transfers = 138, Max value = 16, Average = 0
2   Milliseconds summarizing B2B Integrator/Sterling File Gateway transfers = 0, Max value = 0, Average = 0
```

```
2   Milliseconds summarizing Sterling Connect:Direct transfers = 138, Max value = 16, Average = 0
2a  Milliseconds handling tag mapping for All transfers = 31, Max value = 16, Average = 0
2a1 Milliseconds handling tag mapping for B2B Integrator/Sterling File Gateway transfers = 0, Max value = 0, Average = 0
2a1 Milliseconds handling tag mapping for Sterling Connect:Direct transfers = 31, Max value = 16, Average = 0
2b  Milliseconds getting DVG names associated with transfers being summarized = 0, Max value = 0, Average = 0
```

The values shown are ordered by the numbering on the left. When letters are used, e.g. 1a, 1b, 1c, it's typically done because those are a breakdown of the higher value, in this case 1. (1d1, 1d2, 1d3 are breakdown of the value for 1d.)

Always pay attention to the time spent doing database I/O above. Specifically, the times for:

- Milliseconds running query to get processes to be summarized
- Milliseconds in getEvents to execute SQL to retrieve events
- Milliseconds committing database changes to CC_PROCESS and CC_FILE_TRANSFER

The queries to get processes to be summarized run against the CC_PROCESS table. Long times here indicate a lack of maintenance being done to the ICC summary database tables and/or having so much data in them the database server performs poorly. Likewise, for the time spent committing database changes to CC_PROCESS and CC_FILE_TRANSFER.

The SQL used by getEvents to retrieve events runs against the EVENTS table. Long times here could indicate too many days of summary table data being retained. The process number limit for Connect:Direct servers is only 99,999, at which point process numbers begin at one again. When customers retain enough days of data that process numbers on monitored Connect:Direct servers repeat, it can cause long summary times, and increased memory usage, since when ICC retrieves all events associated with processes it is not able to distinguish between unique instances of them when their process names and numbers are identical, which results in ICC combining the events for multiple processes and longer than normal summary times.

You should also compare the metric values for 1d1, Milliseconds in getEvents to execute SQL to 1d2, Milliseconds in getEvents to construct events. If 1d2 is larger than 1d1 it's an indication that either the amount of memory, or lack thereof, allocated to ICC, or the processor/CPU for the server on which ICC is running, is limiting ICC performance.

TIP - There are queries (see below) that may be run to ascertain the counts of processes to be summarized and the counts of processes that have been summarized.

Processes yet to be summarized:

- `SELECT COUNT(*) FROM CC_PROCESS WHERE PROC_STATUS IN ('upgraded', 'ended', 'user completed ns')`
- `SELECT STARTED_DAY, COUNT(*) FROM CC_PROCESS WHERE PROC_STATUS IN ('upgraded', 'ended', 'user completed ns') group by STARTED_DAY order by STARTED_DAY`

Processes summarized:

- `SELECT COUNT(*) FROM CC_PROCESS WHERE PROC_STATUS IN ('failed', 'success', 'user completed', 'errors')`
- `SELECT STARTED_DAY, COUNT(*) FROM CC_PROCESS WHERE PROC_STATUS IN ('failed', 'success', 'user completed', 'errors') group by STARTED_DAY order by STARTED_DAY`

There are engine properties that may be used to affect the performance of the ProcessSummaryService logic including:

- PROCESS_SUMMARY_THREADS – Default is 4.
- PROCS_TO_SUMMARIZE_AT_ONCE – Default is 50. Max allowed is 255.

- MAX_PROCESSES_TO_SUMMARIZE_AT_ONCE[5] – default is 5000.

In theory, the larger the value for PROCESS_SUMMARY_THREADS the better the ProcessSummaryService logic will perform but actual performance depends on the server ICC is running on and the amount of memory allocated to it. The larger the thread count, the higher the memory requirements will be. Likewise, for the number of processes summarized at once.

TIP – Of the two engine properties that may be used to affect performance of the ProcessSummaryService, try increasing the PROCESS_SUMMARY_THREADS value first.

MAX_PROCESSES_TO_SUMMARIZE_AT_ONCE limits the query used to find processes ready to be summarized at one time.

[5] Introduced in ICC 6.1.2.1

7 Monitoring Servers

Server status and activity are monitored by ICC either by polling servers or by monitored servers pushing events to it. There are two types of monitored servers in ICC:

- OSA and
- Non-OSA

All OSA server types – a relatively new server type for ICC, by definition, push events related to their activity to ICC, while non-OSA servers are typically polled for activity that has occurred since the last time they were contacted.

Server types that are polled include:

- Connect:Direct
- B2Bi/Sterling File Gateway
- Connect:Express
- Sterling Secure Proxy

Server types that push events to ICC include:

- ITXA
- Global Mailbox
- FTP z/OS

ITXA and Global Mailbox are OSA servers and their monitoring is handled by the EventProcessorService and the UnprocessedEventService within ICC. Each ICC EP has one EventProcessorService and one UnprocessedEventService to handle monitoring for all OSA servers assigned to it. Non-OSA server types have a service within ICC known as a NodeService that handles monitoring of their activity and status. There is a CDNodeService for Connect:Direct servers, a GISNodeService for B2Bi servers, etc. And unlike OSA servers, there is one NodeService running in an EP for each non-OSA server assigned to it. I.e. if an EP has 500 non-OSA servers assigned to it there will be 500 NodeServices running in that EP while an EP will only have one EventProcessorService and UnprocessedEventService regardless of how many OSA servers are assigned to it.

7.1 OSA Servers

As previously mentioned, OSA servers push events related to their activity and status to ICC and there are two services in ICC utilized to handle processing of OSA server data, the EventProcessorService and the UnprocessedEventService. Performance of these services is controlled by a combination of:

- Configuration file data in EventProcessorService.xml and
- Engine property values.

OSA event processing may be performed synchronously or asynchronously (depending on the version of ICC you're running), and asynchronous OSA event processing is controlled by various engine property values.

Originally only synchronous processing was supported, and it is currently the default – it's not clear that asynchronous processing performs better than synchronous processing, and the configuration parameters stored in the EventProcessorService.xml data, which is kept in the CC_FILES database table, controlled all aspects of OSA server processing. The parameters in EventProcessorService.xml that affect processing are:

- `hbCheckFrequency`

- `historyCheckFrequency`
- `eventsToProcessAtOnce`

The following table includes the default values for these configuration parameters and explains how they're used:

Parameter	Details	Default
`hbCheckFrequency`	A value in milliseconds that controls the frequency at which the EventProcessorService checks to see which OSA servers are late in sending in Heartbeat events, which is the indicator that they are down. The larger this value is, the longer the delay could be for ICC to ascertain that a down OSA server is down.	10000
`historyCheckFrequency`	A value in milliseconds that controls the frequency at which the EventProcessorService checks for new OSA events to process from servers monitored by this EP. Note when the number of OSA events processed matches `eventsToProcessAtOnce` no wait is performed between checks for new OSA events to process. This is referred to as catchup mode. This property is only used when OSA event processing is done synchronously.	10000
`eventsToProcessAtOnce`	Specifies the limit on OSA records to be retrieved by the UnprocessedEventService from the CC_UNPROCESSED_EVENT table. This property is only used when OSA event processing is done synchronously.	1000

Since OSA servers were introduced to ICC, additional engine properties may now be used to also affect, and in some cases override, the performance of OSA server monitoring. These engine properties are:

- MULTI_THREAD_OSA_EVENT_PROCESSING
- OSA_EVENTS_TO_GET_AT_ONCE_COUNT
- OSA_EVENTS_TO_PROCESS_QUEUE_SIZE
- TIME_BETWEEN_PROCESSING_QUEUED_OSA_EVENTS
- TIME_BETWEEN_GETTING_UNPROCESSED_OSA_EVENTS

The following table includes the default values for these engine properties and explains how they're used:

Property	Details	Default
MULTI_THREAD_OSA_EVENT_PROCESSING	Controls whether OSA event processing is done in a synchronous fashion, where by OSA records received are retrieved from the database, then processed, and then deleted from the database, or in an asynchronous fashion, where one thread – handleUnprocessedEvents, is used to process OSA records previously retrieved from the database by another thread – queueUnprocessedEvents, which retrieves OSA records from the database and queues them for processing by handleUnprocessedEvents.	FALSE
OSA_EVENTS_TO_GET_AT_ONCE_COUNT	Sets the limit used in queries initiated by the queueUnprocessedEvents thread to retrieve unprocessed OSA events. This property is only used when OSA event processing is done asynchronously.	500
OSA_EVENTS_TO_PROCESS_QUEUE_SIZE	Sets the limit on the queue used to contain unprocessed OSA events retrieved by queueUnprocessedEvents thread, which are subsequently processed by the handleUnprocessedEvents thread. Always make the value for this property be some multiple, greater than one, of the OSA_EVENTS_TO_GET_AT_ONCE_COUNT value. This property is only used when OSA event processing is done asynchronously.	5000
TIME_BETWEEN_PROCESSING_QUEUED_OSA_EVENTS	A value in milliseconds that dictates how long the handleUnprocessedEvents thread waits after processing the previous batch of OSA events.	5000

Property	Details	Default
	Note when the batch size of OSA events previously processed matches the OSA_EVENTS_TO_PROCESS_QUEUE_SIZE size, no wait is performed. This is referred to as catchup mode. This property is only used when OSA event processing is done asynchronously.	
TIME_BETWEEN_GETTING_UNPROCESSED_OSA_EVENTS	A value in milliseconds that dictates how long the queueUnprocessedEvents thread waits after retrieving and queuing unprocessed OSA events. Note when the number of OSA events previously retrieved matches the OSA_EVENTS_TO_GET_AT_ONCE_COUNT value no wait is performed. This is referred to as catchup mode. This property is only used when OSA event processing is done asynchronously.	5000

7.2 *Non-OSA Servers*

To limit the resources consumed by each NodeService a configuration parameter named `simultaneousCdPollers` in the `CCEngineService` configuration data, whose default value is 7, is used. (Don't worry about the name of the parameter, it is used by all NodeServices, not just the CDNodeService.)

TIP – For optimum performance `simultaneousCdPollers` is typically set to 25% of the number of total number of polled monitored server types assigned to each EP with a maximum value of 30. This is just a guideline and not definitive however.

There are events generated to help you know when monitoring is not occurring at the expected rate for servers that are polled:

```
CCNS018E Monitor rate out of compliance.  Server ID: {0}  Last poll Date/time:
{1}  Monitor Rest Time: {2}
```

```
CCNS029I Monitor rate back in compliance.  Server ID: {0}  Last poll date/time:
{1}  Monitor Rest Time: {2}
```

TIP – Enable the built-in rule "Monitoring rate out of compliance", which uses these events in a linked rule to know when polling rate of any non-OSA server is not occurring at the expected rate for over an hour. This rule is often triggered because `simultaneousCdPollers` is set too low for activity being monitored and/or the number of monitored servers is too high.

7.2.1 *Connect:Direct*

For best performance, NodeService threads for Connect:Direct servers establish connections to their respective monitored servers once, and do not disconnect those connections after each poll unless an error occurs. When monitoring more than a few thousand Connect:Direct servers though it would be possible for ICC to attempt to exceed the number of socket connections allowed by the server's operating system the EP runs on.

In ICC 6.1.2.1 the engine property `NON_PERSISTENT_CD_CONNECTION` was introduced to avoid exceeding allowed socket connections. By setting this property to true, its default is false, Connect:Direct NodeServices will do a disconnect after each poll cycle and reestablish the connection to their Connect:Direct server at the beginning of each poll cycle.

Unless exceeding the number of socket connections is a problem for one, or more, of your EPs, you should not change the default setting for `NON_PERSISTENT_CD_CONNECTION` from false to true.

7.2.2 *Connect:Direct File Agent*

The FileAgentService, which runs only in the CEP, is responsible for handling monitoring of Connect:Direct File Agents associated with monitored Connect:Direct servers. Monitoring of File Agents is achieved in the FileAgentService by processing the SNMP traps they send that contain their status and activity details.

In ICC 6.1.2.1 the following engine properties were introduced:

- `FILEAGENT_MONITOR_RATE` – default 60
- `MAX_TIME_TO_BEGIN_TRAP_PROCESSING_WITHOUT_WARNING` – default 30000
- `SHOW_FILE_AGENTS`

FILEAGENT_MONITOR_RATE dictates the number of seconds between checks the ICC FileAgentService performs to ensure all monitored Connect:Direct File Agents have sent traps within their specified heart beat interval, which let ICC know they are still running.

The larger the value for FILEAGENT_MONITOR_RATE is, the less time the FileAgentService spends checking to ensure traps were received on-time from File Agents, leaving more time for other ICC processes and threads, thus making ICC more efficient. On the other hand, the larger the value for FILEAGENT_MONITOR_RATE is, the longer it will take for ICC to determine a Connect:Direct File Agent has stopped running. Unless you're monitoring tens of thousands of File Agents, the default value for this property should be fine.

The property MAX_TIME_TO_BEGIN_TRAP_PROCESSING_WITHOUT_WARNING, a number of milliseconds with a default of 30000, tells ICC to generate a warning message in the log files if the time at which an SNMP trap received from a monitored Connect:Direct File Agent, and the time at which the ICC FileAgentService begins processing that trap, is exceeded. These warnings contain the following text:

- `FileAgentService took an inordinate amount of time to begin processing of trap - <delay> ms Trap: <trap data>`

The existence of these messages in the EP log files indicates performance problems are occurring within ICC and you may need to increase the heartbeat interval your Connect:Direct File Agents are configured with, as well as increase the ICC engine property value for FILEAGENT_MONITOR_RATE, to avoid them.

Lastly, the engine property SHOW_FILE_AGENTS, controls whether the status of monitored Connect:Direct File Agents should be displayed in the ICC Swing console. By default, SHOW_FILE_AGENTS is true if the number of monitored servers is less than the value of another engine property, TREE_NODE_SIZE_LIMIT, and false otherwise.

Note TREE_NODE_SIZE_LIMIT was also introduced in ICC 6.1.2.1 and its default is 5000.

Setting SHOW_FILE_AGENTS true will cause the status of File Agents to be displayed in the Swing console regardless of the number of monitored servers. Note displaying the status of File Agents in the Swing console is very CPU intensive, and slows performance of the ICC Swing console, so you are advised not to set the property SHOW_FILE_AGENTS to anything other than false.

Note the ICC Web console handles display of monitored Connect:Direct File Agents well without the performance issues that will occur within the ICC Swing console.

*7.2.3 **Sterling File Gateway and B2Bi***

Monitoring B2Bi servers may be very problematic from a performance standpoint and difficult to troubleshoot as there are multiple factors involved that could cause issues, including:

- Poor B2Bi database performance
- Poor ICC database performance
- Simply too much activity to monitor

TIP – For best performance do not run B2Bi on the same server that runs ICC.

TIP – For best performance do not use the same database server for B2Bi and ICC.

The first place to look when performance problems are reported is the B2Bi metrics log file contents. There you will find output, generated once an hour for each monitored B2Bi server, by the three main threads in a GISNodeService used to monitor B2Bi servers:

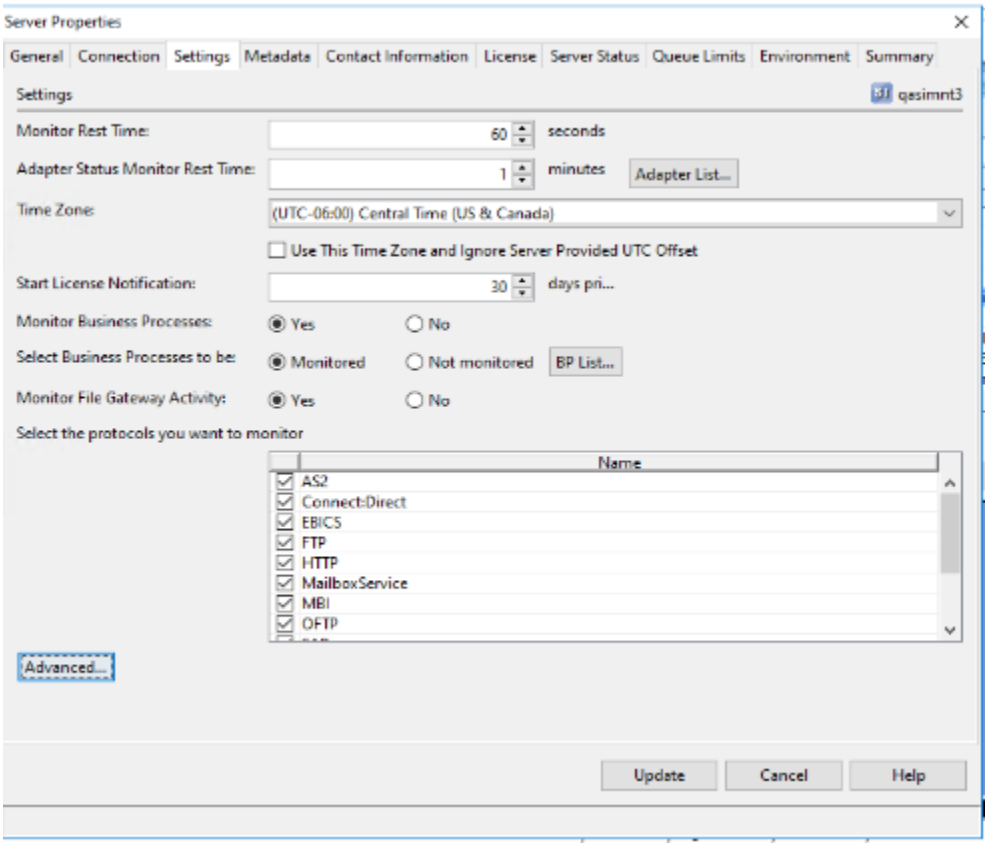
- GetAftHistoryTask – used for monitoring the “advanced file transfer” data, also known as protocols
- GetBpHistoryTask – used for monitoring business process activity and
- GetFgHistoryTask – used for monitoring Sterling Filegateway activity

These threads are responsible for:

- Converting the data received into Control Center events, aka SCCNotifications
- Doing “pre-rule” processing – i.e. looking for a rule match so the RuleService does not have to
- Storing the events generated in the `EVENTS`, `EVENTS_EXT` and appropriate `_STATS_LOG` tables
- Notifying the ProcessSummaryService about processes seen to have started and ended
- Passing events on to the RuleService for processing

Note the GetAftHistoryTask thread also handles the retrieval of other information from the B2Bi server, such as adapter status information, in addition to its primary function – getting protocol activity data.

TIP – Monitoring the adapter status on B2Bi servers is an onerous process and has a negative impact on performance. The default “Monitor Rest Time” for updating the status of B2Bi adapters is one minute. You can lessen the impact on performance of B2Bi monitoring by increasing this value, which is done via the monitored server’s Server Properties Settings dialog.



Each of these three main threads log metrics, once an hour, for each B2Bi server monitored by ICC. When troubleshooting performance issues related to the monitoring of B2Bi servers, always start by looking at the "high level", summary messages, output to the B2Bi metrics log file. E.g.

- `14 Jan 2019 17:00:00,006 629602 [qasimnt3(3)] INFO B2BiMetrics - CGIS007I AFT Avg ms/req: 508, Avg # records/req: 2, # recs: 23, # reqs: 11 (1 in catchup), Avg ms to build events/req: 27, Avg events/req: 7, Avg log ms: 117, Avg # events/batch: 39, # events: 78`
- `14 Jan 2019 17:00:00,006 629602 [qasimnt3(3)] INFO B2BiMetrics - CGIS008I SFG Avg ms/req: 714, Avg # records/req: 82, # recs: 909, # reqs: 11 (1 in catchup), Avg ms to build events/req: 42, Avg events/req: 27, Avg log ms: 296, Avg # events/batch: 74, # events: 298`
- `14 Jan 2019 17:00:00,006 629602 [qasimnt3(3)] INFO B2BiMetrics - CGIS009I BP Avg ms/req: 2362, Avg # records/req: 3473, # recs: 79886, # reqs: 23 (17 in catchup), Avg ms to build events/req: 3028, Avg events/req: 7021, Avg log ms: 55, Avg # events/batch: 98, # events: 161496`

Note each of the summary messages above indicate the identity of the monitored server they're related to inside the square brackets ("[]") that follow the time stamp for each log message.

For performance issues related to:

- Protocol data collection, which is rarely true, focus on the `CGIS007I` message data, and the output that follows it.
- Sterling Filegateway data collection, focus on the `CGIS008I` message data, and the output that follows it
- Business process data collection, focus on the `CGIS009I` message data, and the output that follows it

Each of the summary messages provide a wealth of information that can quickly help isolate where to look for problems, if not to identify what they are. Here are details on each value they contain:

What	Description
`Avg ms/req`	Response time for requests to the B2Bi server for data. The response time is affected by the number of records received per request, but it should never exceed 20 seconds and it should be much lower – sub 10 seconds. Long request response times are typically due to unmaintained B2Bi database tables and/or indices.
`Avg # records/req`	Average number of records received per request by ICC. When B2Bi servers are busy, this number should match, or be close to, the limit specified for requests in ICC, which comes either from the `CCEngineService` configuration value for `recordLimit`, or `siRecordLimit` (when specified). Note these configuration values are settable via the ICC Web Console. The default value for `siRecordLimit` is 4999. Better performance is obtained from busy B2Bi servers when it is set higher. 20000 is the recommended value for `siRecordLimit` to use but increasing the value beyond the default of 4999 may require changes to B2Bi. Always consult support prior to increasing `siRecordLimit` beyond its default value.
`# recs`	Total number of records received from B2Bi for the hour reported on.
`# reqs`	Total number of requests made by ICC for data from B2Bi during the hour reported on. Following the number of requests, you may see a "catchup" count also. E.g. `(17 in catchup)` When the catchup count matches, or is a large percentage, of the number of requests made, it's an indication that ICC was unable to keep up with the level of activity on the B2Bi server during the hour reported. ICC is typically unable to "keep up" because of long response times, but it may be due to other factors including the time to insert events into the ICC database, or the time ICC takes to build events to be inserted.
`Avg ms to build events/req`	Average time ICC takes to construct events from data received from B2Bi. When this number is large it can cause ICC to fall behind in monitoring B2Bi.

What	Description
`Avg events/req`	Average number of events generated from data received per request to B2Bi.
`Avg log ms`	Average time ICC took to add the events it created to the ICC database from the B2Bi data it received. When this number is large it can cause ICC to fall behind in monitoring B2Bi.
`Avg # events/batch`	Average number of events inserted per SQL INSERT initiated by ICC to add the events it created to the ICC database.
`# events`	Total number of events generated by ICC from the data received from B2Bi during the hour reported on. You can use this value to ascertain the times of heavy loads on the B2Bi server.

Each summary log message is followed by details like this:

```
15 Jan 2019 10:00:07,205 61836801 [qasimnt3(3)] INFO  B2BiMetrics - Metrics for the current hour
        Total time to get data: 29877 ms
        Last request to get data took: 687 ms  (&lt; 5000 ms is preferable)
        Maximum time to get data took: 875 ms  (&lt; 5000 ms is preferable)
        Average time to get data: 497 ms  (&lt; 5000 ms is preferable)
        Total number of requests for data: 60
        Requests for data in catch up mode: 0
        Events created: 297
        Total time to log events: 110 ms
        Time to log data for data last received: 16 ms  (&lt; 3000 ms is preferable)
        Maximum time to log data: 32 ms  (&lt; 3000 ms is preferable)
        Average time to log events: 27 ms  (&lt; 3000 ms is preferable)
        Total events logged: 4
        Total time to get and process data: 30267 ms
        Time to get and process data for last request: 703 ms  (&lt; 10000 ms is preferable)
        Maximum time to get and process data for a request: 875 ms  (&lt; 10000 ms is preferable)
        Average time to process data received: 4 ms  (&lt; 5000 ms is preferable)
        Total records received: 1094
        Records received for last request: 0
        Maximum records received for one request: 504
        Average number of records received per request: 18
        Average number of events created per request: 4
        Average number of events logged per request: 74
        Additional values (for internal use)
                AbstractHistoryHandler.logEventToDB: 110 ms
                AbstractHistoryHandler.logStatToDB: 63 ms
                AbstractHistoryHandler.processStatRecords.logEventToDB: 110 ms
                AddStat.a: 16 ms
                AddStat.b: 0 ms
                BuildEventFromStatData: 15 ms
                FgProcessRecord: 16 ms
                FgProcessRecord.a: 0 ms
                FgProcessRecord.b: 16 ms
                GetHistory.a: 29939 ms
                GetHistory.b: 32 ms
                GetHistory.c: 0 ms
                GetResults.a: 0 ms
                GetResults.b: 29877 ms
                GetResults.c: 0 ms
                GetResults.d: 0 ms
                GetResults.e: 31 ms
                GetResults.f: 0 ms
                GetResults.g: 31 ms
                GetResults.h: 0 ms
                GetResults.i: 0 ms
                GetResults.j: 0 ms
                GetResults.k: 0 ms
                MergeEventResults: 0 ms
                MergeEventResults.sort: 0 ms
                NodeService.logEventToDB.insertNotifications: 47 ms
                NodeService.logEventToDB.rulePreProcessing: 0 ms
                ProcessData.a: 0 ms
                ProcessData.b: 29971 ms
                ProcessData.c: 0 ms
                ProcessData.d: 141 ms
                ProcessData.e: 0 ms
                ProcessStatRecord.a: 0 ms
                ProcessStatRecord.b: 16 ms
                ProcessStatRecords.a: 16 ms
                ProcessStatRecords.b: 172 ms
                SCCNotif.insertNotification: 16 ms
                SCCNotif.insertNotification-list: 47 ms
                SCCNotif.insertNotification-list.executeBatchUpdate: 47 ms
                SCCNotif.insertNotification-list.insertNotificationExt: 0 ms
                SCCNotif.insertNotification-list.keepProcessSummaryServiceInformed: 0 ms
                SCCNotif.insertNotification.executeUpdate: 16 ms
                SCCNotif.insertNotification.keepProcessSummaryServiceInformed: 0 ms
                SCCNotificationJdbcDAO.keepProcessSummaryServiceInformed: 0 ms
```

This information is basically a rehash of the summary data, with some embellishments. The “additional values (for internal use), which can be very helpful to further pin point areas of performance problems but can be difficult to decipher.

The following outline form of the data shown may help as indentation is used to give a breakdown of the higher-level's time value (note the order of the outline does not always indicate the order of the work performed).

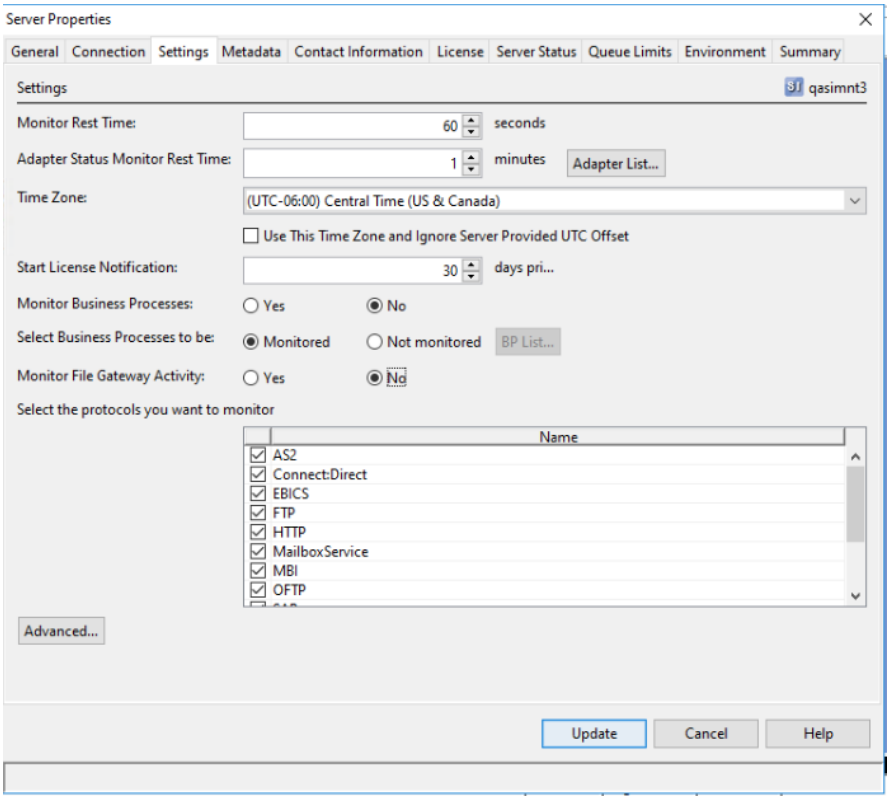
- ProcessData.a – time to invoke adjustToNodeTime() which normalizes time values
- ProcessData.b – time to invoke getHistory()
    - GetHistory.a – time to get data from server and to build stat records out of it
        - GetResults.a – time to construct request for data
        - GetResults.b – time to get response as a document and convert it to a String[1]
        - GetResults.c – time to build a list of Arrived file records
        - GetResults.d – time to invoke processArrivedFileRecords()
        - GetResults.e – time to build a list of Route records
        - GetResults.f – time to invoke processRouteRecords()
        - GetResults.g – time to build a list of Delivery records
        - GetResults.h – time to invoke processDeliveryRecords()
        - GetResults.i – time to build a list of Route records
        - GetResults.j – time to deduplicate the list of Route records
        - GetResults.k – time to get and process missing information
    - GetHistory.b – time to process all the stat records constructed
        - FgProcessRecord.a – time to invoke getProcessName()
        - FgProcessRecord.b[2]
            - AddStat.a – time to construct a GISStatistic
            - AddStat.b – time to put the GISStatistic built in appropriate data structure
    - GetHistory.c – time to order the collection of records constructed
- ProcessData.c – time to adjust times and set sequence numbers for getHistory() results
- ProcessData.d – time to invoke processStatRecords() which logs and processes getHistory() results
    - ProcessStatRecords.a – time to skip duplicate data
        - ProcessStatRecord.a – time to construct SCCNotification
        - ProcessStatRecord.b – time to attempt insertion of notification
            - SCCNotif.insertNotification
                - SCCNotif.insertNotification.executeUpdate
                - SCCNotif.insertNotification.insertNotificationExt
                - SCCNotif.insertNotification.keepProcessSummaryServiceInformed
    - ProcessStatRecords.b - time to process the none duplicate data
        - BuildEventFromStatData – time to construct SCCNotifications for non-dupe data
        - AbstractHistoryHandler.processStatRecords.logEventToDB
            - AbstractHistoryHandler.logStatToDB - time to update _STAT_LOG

        - AbstractHistoryHandler.logEventToDB - time to update EVENTS and more…
            - NodeService.logEventToDB.rulePreProcessing
            - NodeService.logEventToDB.insertNotifications
                - SCCNotif.insertNotification-list
                    - SCCNotif.insertNotification-list.executeBatchUpdate
                    - SCCNotif.insertNotification-list.insertNotificationExt
                    - SCCNotif.insertNotification-list.keepProcessSummaryServiceInformed
    - Unaccounted for time is time to queue events for broadcasting
- ProcessData.e – time to update checkpoint data

[1] To see just the amount of time to get the response, look for "ms to perform and extractFirstAttachment for" in the Engine log files
[2] Equivalent to FgProcessRecord time

TIP – Better B2Bi monitoring performance may be obtained by limiting the protocols specified for monitoring to the ones you need data for.

TIP – Better B2Bi monitoring performance may be obtained by disabling collection of Sterling File Gateway and / or Business Process data.



8 Rules and Actions

As ICC events are generated, they are processed by the RuleService in the EP they originated from, and the number of rules defined, and the actions associated with rules that are triggered, can have a large impact on the performance of ICC.

TIP – Disable or delete all ICC rules, including built-in rules, that are not being used to improve ICC performance.

ICC can typically handle having 1000s of enabled rules without incurring performance problems, but while some actions, e.g. email actions, have little impact on ICC performance, others can have a large, negative, impact, e.g. OS Commands and Alert 0 actions.

8.1 OS Commands

OS Commands specify a script, or batch file, ICC is to perform when a rule is triggered by an event. When a script or batch file initiated by ICC takes a long time to run it delays processing of other events by ICC so be sure they always run quickly.[6]

TIP – When scripts or batch files will take more than a few milli-seconds to run, have them execute whatever they do asynchronously, instead of synchronously, when possible.

8.2 Alert 0

Alert 0 is a special alert action that tells ICC to find, and handle, any alerts that are "related" to the event that triggered that rule. To find alerts related to an event, ICC runs one, or more, SQL statements and that SQL causes contention on the ICC EVENTS table and can result in noticeable performance issues.

TIP – Since alerts can only be related to another when they are generated by the same SLC instance or by a linked Rule, Alert 0 actions should only be specified in conjunction with linked rules, or a rule whose criteria matches SLC events. Beware that some built-in rules may violate this principle!

Erroneous Alert 0 actions are something to look for whenever you notice event broadcasting performance is slow, which occurs in the Rule service, the EventMonitorService, and more. In most cases where Alert 0 actions are used erroneously, the "No operation" action should be used instead.

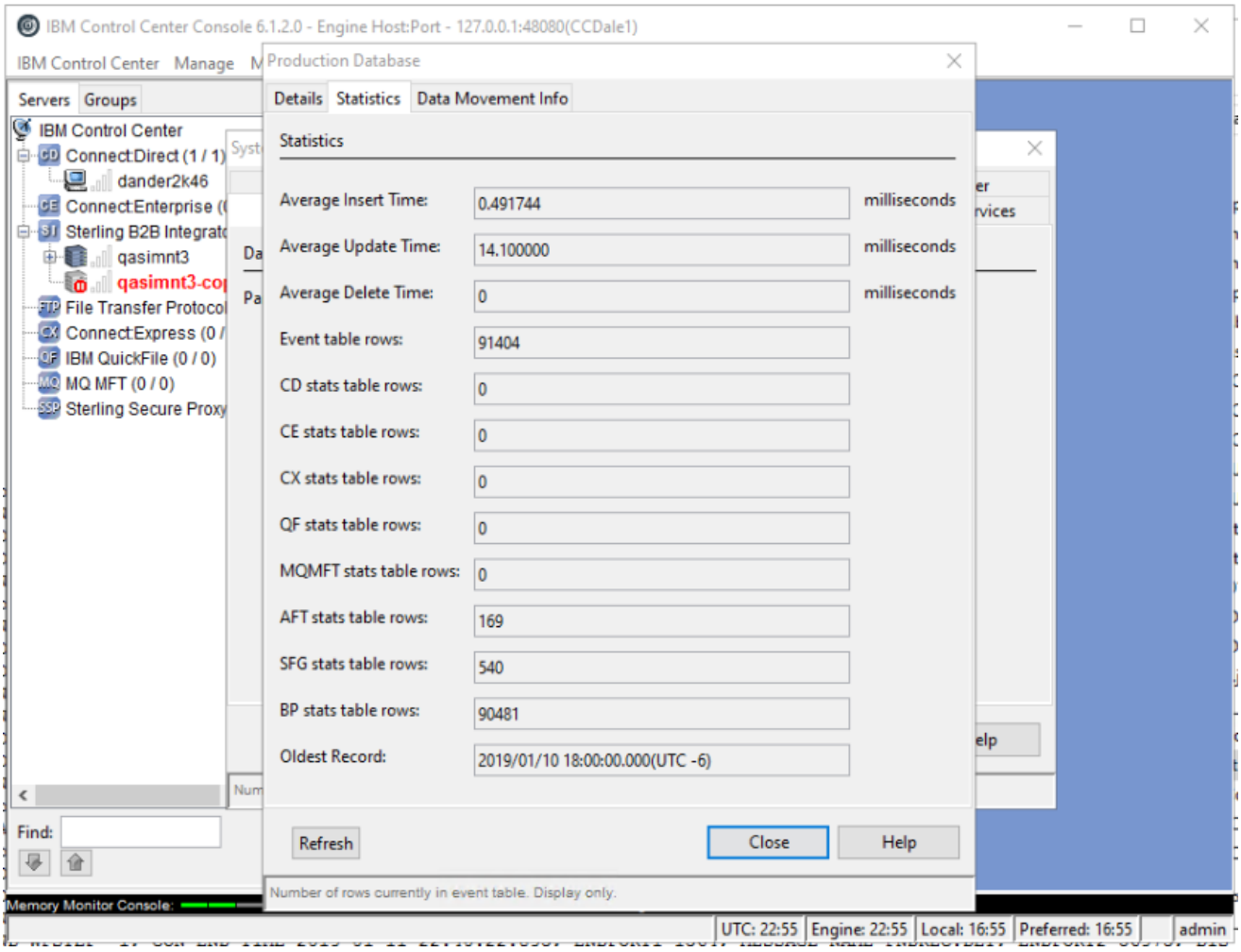
[6] When a script or batch file is initiated by ICC an engine log entry is written that tells how long it took to run.

9 Miscellaneous

9.1 Row Counts

At startup, ICC initiates a query to ascertain the row counts in the various database tables used to hold event data. These queries can be onerous to performance as they may cause contention on those tables while ICC is trying to add data to them.

These row counts are displayed by the ICC Swing console:



If you do not mind having the counts above be wrong, and only show the amount of data added since ICC was last started, the engine property `DO_NOT_CALCULATE_ROW_COUNTS` may, and should, be specified with a value of TRUE – the default value is FALSE.

When `DO_NOT_CALCULATE_ROW_COUNTS` is set to TRUE ICC will no longer initiate queries to get row counts for its tables at startup or when the daily purge operation is run.

9.2 ***ICC Web Console***

The ICC Web console performance is highly dependent on the performance of the database server. When queries initiated by the web console run slowly, the time to show data in the web console will be long.

By default, the ICC Web console typically displays a years' worth of information when displaying data about monitored servers, including alert, completed process, and completed file transfer counts. The queries used to ascertain this information may be too onerous for the ICC database server, which will result in long delays in displaying the results.

Sometimes these performance problems may be addressed by doing index maintenance and / or retaining less data in the ICC database tables.

TIP – An engine property named `DEFAULT_DAYS_OF_HISTORY`, whose default is 366, may be used to limit the date range used by the queries to less (or potentially more) than a years' worth of information, which can result in far better performance.

To see the queries initiated by the Web console, and the amount of time it takes to run them, look at the contents of the ICC web server log files. They can be found in:

`<CC Install folder>/web/wlp/usr/servers/defaultServer/logs`

Either looking at the `messages.log` or `trace.log` file will work. If you do not see queries, and the times it takes to perform them, in either of these files, update the file `jvm.options`, which is found in the `defaultServer` folder, such that the values are `TRUE` for both:

- `10x.logDebugMessagesToStandardOut and`
- `10x.printStackTracesToStandardErr`

You may also need to update the file:

`<CC Install folder>/web/wlp/usr/servers/defaultServer/log4j2.xml`

The bottom of this file needs to be changed from:

```
<!--         <Root level="DEBUG" includeLocation="false" additivity="false"> -->
<!--             <AppenderRef ref="textFile" /> -->
<!--             <AppenderRef ref="Console" level="DEBUG"/> -->
<!--         </Root> -->

        <Root level="INFO" includeLocation="false" additivity="false">
            <AppenderRef ref="textFile" />
            <AppenderRef ref="Console" level="ERROR"/>
        </Root>

      </Loggers>
</Configuration>
```

To:

```
<!--         <Root level="DEBUG" includeLocation="false" additivity="false"> -->
<!--             <AppenderRef ref="textFile" /> -->
<!--             <AppenderRef ref="Console" level="DEBUG"/> -->
<!--         </Root> -->

        <Root level="DEBUG" includeLocation="false" additivity="false">
            <AppenderRef ref="textFile" />
            <AppenderRef ref="Console" level="DEBUG"/>
        </Root>

      </Loggers>
</Configuration>
```

After a few moments, look at new output that is logged. Restarting the ICC web server should not be necessary for these changes to take effect.

TIP – When you enable logging of debug messages it is best if you limit web console usage to a single user to prevent so much output being logged that you cannot find what you are looking for.

Also, know time stamp values logged may be in UTC, or local time, or a combination of the two.

9.3 Clearing Queued Processes

There is a thread in ICC named QueuedProcessesClearJob whose purpose is ensuring processes other than from Connect:Direct that have completed do not remain in the Web console's queued processes view after they have completed, which occasional occurs when additional events related to a process are received after its end event.

An engine property named QUEUED_PROCS_CLEAR_JOB_INTERVAL, whose default is 5 (unit is minute), controls how frequently this thread looks for processes that have completed but are still shown as queued.

TIP – A simple way to lower contention on the ICC CC_PROCESS database table is to decrease the frequency at which the QueuedProcessesClearJob runs, which can be achieved by increasing the value for the engine property QUEUED_PROCS_CLEAR_JOB_INTERVAL to a value greater than five.

9.4 Memory

ICC requires quite a bit of memory to run. Factors involved in knowing how much memory is required include, but are not limited to:

- The number and types of servers being monitored
- The number of Rules and SLCs
- The configuration setting value for simultaneousCdPollers
- The batch size specified for ICC database table operations

It is beyond the scope of this document to tell you how to compute the exact amount of memory you'll need to run ICC, but typically, the more that you can allocate to it the better and don't try anything less than 6GB for ICC, which means no less than having 8GB for the server ICC is installed on. ICC has no known "memory leaks" so if you find your ICC installation crashing due to "out of memory" errors, allocate more and restart it.

If you'd like to know more, please contact sales and ask about having a performance review done. IBM Professional Services would be able to perform a "Server Sizing Study" for you.

9.5 Reporting

Out of the box, ICC limits reports to 5000 rows. You can increase the report limit by updating ICC engine property values. And when you increase the report limit, you may also need to allow more memory to be allocated to Cognos, the ICC reporting service, by updating the cogstartup.xml file, to avoid out of memory errors from occurring.

To increase the report limit, edit the ReportService.xml property reportRecordLimit.

To allocate more memory to Cognos, the Cognos web server JVM heap size to be specific, you will need to edit the cogstartup.xml file found in the ICC installation subfolder Cognos/configuration. By default, the maximum heap size allowed may only be 768mb (newer releases of ICC specify a maximum of 4096mb). Look for the value associated with the parameter "dispatcherMaxMemory" in cogStartup.xml and increase the value found there if out of memory exceptions occur when you attempt to run large reports.

```
    <!-- dispatcherMaxMemory: Specifies the maximum amount of memory in MB that can be used by the process. -->
    <!-- This value determines the amount of memory used by the Java Virtual Machine and depends on how much memory is available. If
    this value is too high, the process will fail to start and no log information will be generated. Invoke the test action to determine if this
    value is valid. -->
<crn:parameter name="dispatcherMaxMemory">
    <crn:value xsi:type="xsd:unsignedInt">4096</crn:value>
</crn:parameter>
```

The engine.properties value for DB_BATCH_SIZE specifies the number of records to use for batch insertion when temporary tables are created for use by Cognos Reports. Valid values are 1 - 2,147,483,647. This value must contain an integer greater than or equal to 1. The default is 1000 rows. Increasing the batch size above the default of 1000 rows might improve the speed of the report by one or two seconds. However, increasing the batch size could result in out of memory errors occurring.

The engine.properties value for `JDBC_DRIVER_MAX_ROWS` controls how many records are read at one time from the database when paging thru tables with large amounts of data for Sterling File Gateway Detail by Producer reports. The default for this property is 10000 and it is only used for this one report. You may get better performance for this report by increasing this value.

10 Performance Related Engine Properties and Configuration Settings

10.1 ***Engine Properties***

Property	Default	Description
BYPASS_EVENT_MONITOR_FOR_EVENTS	True	When true events go directly to the SLC and FileAgent services, which is more efficient, instead of indirectly via the EventMonitor service. May only be set to true for single EP installations.
CLEAR_NODE_STATUS_AT_CEP_STARTUP	True	Tells the CEP to reset the status of all monitored servers to `Unknown` at startup instead of leaving them in their current state, which could cause invalid status information to be shown until all EPs have been started and begun monitoring servers they are the active EP for. Note a message will be logged, and written to the console, telling how long this operation takes when the property is set to true. New in ICC 6.1.2.1
DEFAULT_DAYS_OF_HISTORY	366	Used to limit the date range used by the Web console queries for data to less (or potentially more) than a years' worth of information, which can result in far better performance. Set this property lower for better Web console performance, albeit causing it to show less data.
DO_NOT_CALCULATE_ROW_COUNTS	False	Tells ICC whether at startup, and subsequently once a day, to initiate queries to ascertain the row counts in various database tables it stores information in. The values obtained are only seen in the ICC Swing console. These queries can be onerous to run - they can take a long time and cause database contention. Set this property True for best performance. As of ICC 6.1.2.1 changes made to this property do not require ICC to be restarted to take effect.

Property	Default	Description
EVENT_MONITOR_EVENT_COUNT_FOR_CATCHUP	200	Tells the EventMonitor the count of events that will trigger it to enter catchup mode, which means it will not wait the normal amount of time before attempting to retrieve more event data for processing, which is subsequently passed on to the SLC and FileAgent services. This property is not new in ICC 6.1.2.1, but the previous default was 100. As of ICC 6.1.2.1 changes made to this property do not require ICC to be restarted to take effect.
EVENT_MONITOR_EVENT_LIMIT_PER_RUN	1000	Tells the EventMonitor the query limit for retrieving event data. This property is not new in ICC 6.1.2.1, but the previous defaults were smaller. As of ICC 6.1.2.1 changes made to this property do not require ICC to be restarted to take effect.
EVENT_MONITOR_JUST_SLC	False	When this property is true the EventMonitor logic limits the query used for retrieving event data to only get data the SLC service requires, which makes the query, and the EventMonitor, run faster. For best performance set this property TRUE but you should not do this if Connect:Direct File Agents are being monitored. Changes made to this property do not require ICC to be restarted to take effect. New in ICC 6.1.2.1
EVENT_MONITOR_THREADS	Number of server CPUs	Tells the EventMonitor logic how many threads to use while reconstructing ICC events from database data. Changes made to this property do not require ICC to be restarted to take effect. Default changed in ICC 6.1.2.1 from 2 to the number of CPUs the server running ICC has.

Property	Default	Description
FILEAGENT_MONITOR_RATE	60	Tells how many seconds to wait in-between checks for status of Connect:Direct File agents. When monitoring 1000s of File agents this number should be increased to limit the amount of time spent status checking by the FileAgentService. Changes made to this property do not require ICC to be restarted to take effect. New in ICC 6.1.2.1
MAX_PROCESSES_TO_SUMMARIZE_AT_ONCE	5000	This value limits the number of ready to be summarized processes that will be queried for by the ProcessSummaryService. This property exposes a previously hard-coded value that was already in the logic. Changes made to this property do not require ICC to be restarted to take effect. New in ICC 6.1.2.1
MAX_TIME_TO_BEGIN_TRAP_PROCESSING_WITHOUT_WARNING	30000	A value in milliseconds that tells ICC when it should generate warnings in the ICC log files about slow processing of SNMP traps received from Connect:Direct File agents. New in ICC 6.1.2.1
MULTI_THREAD_OSA_EVENT_PROCESSING	False	Controls whether OSA event processing is done in a synchronous fashion, where by OSA records received are retrieved from the database, then processed, and then deleted from the database, or in an asynchronous fashion, where one thread –handleUnprocessedEvents, is used to process OSA records previously retrieved from the database by another thread – queueUnprocessedEvents, which retrieves OSA records from the database and queues them for processing by handleUnprocessedEvents.
MULTI_THREAD_SERVER_REASSIGNMENT	True	Prior to ICC 6.1.2.1 server reassignments were performed one at a time. I.e. serially. In ICC

Property	Default	Description
		6.1.2.1 multiple threads may be used to speed up the server reassignment process. This property is used in conjunction with the properties SERVER_REASIGNMENT_QUEUE_SIZE and SERVER_REASSIGNMENT_THREADS. For best server reassignment performance this property should be TRUE. Changes made to this property do not require ICC to be restarted to take effect. New in ICC 6.1.2.1
NON_PERSISTENT_CD_CONNECTION	False	Prior to ICC 6.1.2.1 NodeServices for Connect:Direct servers would stay connected, if possible, between polls of their respective servers. While it is less efficient to disconnect between polls, to facilitate monitoring of thousands of servers without running out of server socket connections, you can now configure ICC to do so, which limits the number of socket connections ICC keeps open at one time. Changes made to this property do not require ICC to be restarted to take effect. New in ICC 6.1.2.1
OSA_EVENTS_TO_GET_AT_ONCE_COUNT	500	Sets the limit used in queries initiated by the queueUnprocessedEvents thread to retrieve unprocessed OSA events. This property is only used when OSA event processing is done asynchronously, which is controlled by the MULTI_THREAD_OSA_EVENT_PROCESSING property.
OSA_EVENTS_TO_PROCESS_QUEUE_SIZE	5000	Sets the limit on the queue used to contain unprocessed OSA events retrieved by queueUnprocessedEvents thread, which are subsequently processed by the handleUnprocessedEvents thread.

Property	Default	Description
		Always make the value for this property be some multiple, greater than one, of the OSA_EVENTS_TO_GET_AT_ONCE_COUNT value. This property is only used when OSA event processing is done asynchronously, which is controlled by the MULTI_THREAD_OSA_EVENT_PROCESSING property.
PROCESS_SUMMARY_THREADS	4	Dictates how many threads are used by the ProcessSummarizationService to summarize completed processes. The larger this value is set; the higher memory and CPU utilization will become.
PROCS_TO_SUMMARIZE_AT_ONCE	50	Dictates how many processes may be summarized at once by each process summarization thread, which are called Process Summary Workers. Changes made to this property do not require ICC to be restarted to take effect. The larger this value is set; the higher memory utilization will become. Changes made to this property do not require ICC to be restarted to take effect.
QUEUED_PROCS_CLEAR_JOB_INTERVAL	5	Controls the frequency at which the QueuedProcessesClearJob runs, which ensures completed B2Bi processes are removed from the Queued process view. The larger this value is made the less contention there will be on the CC_PROCESS database table, and the longer completed B2Bi processes may remain in the Web console queued process view. Set this property value higher for better performance.
SERVER_REASSIGNMENT_QUEUE_SIZE	100000	Count of server reassignments that may be queued for handling. Note this property is used in conjunction with

Property	Default	Description
		the property MULTI_THREAD_SERVER_REASSIGNMENT. New in ICC 6.1.2.1
SERVER_REASSIGNMENT_THREADS	Number of server CPUs	Prior to ICC 6.1.2.1, server reassignments were handled synchronously. I.e. one at a time. Multiple threads may be configured as of ICC 6.1.2.1 to handle reassignments and will be by default. Note this property is used in conjunction with the property MULTI_THREAD_SERVER_REASSIGNMENT. New in ICC 6.1.2.1
SERVER_STARTUP_SHUTDOWN_THREADS	Number of server CPUs	Prior to ICC 6.1.2.1 NodeService threads for monitor servers were started one at a time when EPs started, and they were stopped one at a time when EPs were shutdown. As of ICC 6.1.2.1 multiple threads may be used to speed up the startup and shutdown processes. This property is used in conjunction with the property USE_MULTIPLE_THREADS_FOR_NODESERVICE_STARTUP_AND_SHUTDOWN. New in ICC 6.1.2.1
SHOW_FILE_AGENTS	True when number of monitored servers less than TREE_NODE_SIZE_LIMIT property value.	Controls whether or not Connect:Direct File agents will be displayed in the Swing console node tree. Displaying File agents in the Swing console's node tree is very processor intensive due to the number of status events that must be handled and has a large, detrimental effect to Swing console performance. Note you are always able to efficiently view Connect:Direct File agents and their status via the ICC Web console. By default, File agents will not be displayed when the number of monitored servers exceeds the

Property	Default	Description
		property value for TREE_NODE_SIZE_LIMIT. For best performance do not alter this property. New in ICC 6.1.2.1
SUMMARIZE_MILLIS_DELAY_FOR_PROCS	1000	This property value, a value in milliseconds, limits the query used by the ProcessSummaryService when searching for processes ready to be summarized. Processes that completed less than the amount of time specified will not be returned. This is not a new property, but the default was changed from 0 to 1000 in ICC 6.1.2.1 to avoid occasional problems detected during testing that caused events to not be associated with completed processes when they should have been. Changes made to this property do not require ICC to be restarted to take effect.
SUMMARY_TABLES_PURGE_BATCH_SIZE	100	Used as a row limit in the SQL used to delete summary table data. Since the SQL used to delete data causes more contention on the database than the SQL used to find the data to be deleted, the value for this property should be kept relatively small compared to the property SUMMARY_TABLES_PURGE_QUEUE_SIZE.
SUMMARY_TABLES_PURGE_FILE_COUNTS	False	Tells ICC to also purge data from the FILE_COUNTS database table when other summary data is purged. For best performance set this property to TRUE.
SUMMARY_TABLES_PURGE_QUEUE_SIZE	100000	Sets the limit on the SQL used to ascertain the data to be deleted from the summary tables.
SUMMARY_TABLES_PURGE_ROLL_UP	False	Tells ICC to also purge data from the ROLL_UP database table when other summary data is purged. For best performance set this property to TRUE.

Property	Default	Description
TIME_BETWEEN_GETTING_UNPROCESSED_OSA_EVENTS	5000	A value in milliseconds that dictates how long the queueUnprocessedEvents thread waits after retrieving and queuing unprocessed OSA events. Note when the number of OSA events previously retrieved matches the OSA_EVENTS_TO_GET_AT_ONCE_COUNT property value no wait is performed. This is referred to as catchup mode. This property is only used when OSA event processing is done asynchronously, which is controlled by the MULTI_THREAD_OSA_EVENT_PROCESSING property.
TIME_BETWEEN_PROCESSING_QUEUED_OSA_EVENTS	5000	A value in milliseconds that dictates how long the handleUnprocessedEvents thread waits after processing the previous batch of OSA events before attempting to process more. Note when the batch size of OSA events previously processed matches the OSA_EVENTS_TO_PROCESS_QUEUE_SIZE size, no wait occurs. This is referred to as catchup mode. This property is only used when OSA event processing is done asynchronously, which is controlled by the MULTI_THREAD_OSA_EVENT_PROCESSING property.
TREE_NODE_SIZE_LIMIT	5000	Tells the Swing console logic to not display the node tree by default when the number of monitored servers exceeds the value specified. A menu option in the Swing console may be used to display the node tree regardless of the number of monitored servers. Note contrary to the message displayed in the Swing console when TREE_NODE_SIZE_LIMIT is exceeded, the number of servers in the node tree, unlike the number of Connect:Direct File agents, has little to no effect on Swing console or EP performance. New in ICC 6.1.2.1

Property	Default	Description
UPDATE_SERVER_GROUPS_NEW_LOGIC	True	Prior to ICC 6.1.2.1 the time to add servers grew by a significant amount as the number of monitored servers increased. Inefficiencies found in the logic used to update server groups were primarily to blame for the slowness. The logic was updated to make updates of server groups significantly faster and this property controls whether the new logic is used. For best performance this property should be TRUE. Changes made to this property do not require ICC to be restarted to take effect. New in ICC 6.1.2.1
USE_MULTIPLE_THREADS_FOR_NODESERVICE_STARTUP_AND_SHUTDOWN	True	Prior to ICC 6.1.2.1 NodeService threads, which monitor servers, were started one at a time when EPs started, and stopped one at a time when EPs were shutdown. Now multiple threads may be used to speed up the ICC startup and shutdown processes. This property is used in conjunction with SERVER_STARTUP_SHUTDOWN_THREADS. For best performance this property should be TRUE. New in ICC 6.1.2.1

10.2 ***Configuration Settings***

Setting	Default	Description	Location
`simultaneousCdPollers`	7	Limits the resources consumed by all NodeServices, not just those for Connect:Direct despite its name, by restricting the number that may communicate with their respective monitored servers simultaneously. For optimum performance `simultaneousCdPollers` is typically set to 25% of the number of total number of polled monitored server types assigned to each EP with a maximum value of 30. This is just a guideline and not definitive however.	`CCEngineService.xml`
`eventsToProcessAtOnce`	1000	Specifies the limit on OSA records to be retrieved by the `UnprocessedEventService` from the `CC_UNPROCESSED_EVENT` table. This property is only used when OSA event processing is done synchronously.	`EventProcessorService.xml`
`hbCheckFrequency`	10000	A value in milliseconds that controls the frequency at which the `EventProcessorService` checks to see which OSA servers are late in sending in Heartbeat events, which is the indicator that they are down. The larger this value is, the longer the delay could be for ICC to ascertain that an OSA server has stopped communicating with ICC and that its status should change to Down. Increasing this property value boosts performance and the expense of learning that OSA servers are down as soon as possible.	`EventProcessorService.xml`
`historyCheckFrequency`	10000	A value in milliseconds that controls the frequency at which the `EventProcessorService` checks for new OSA events to process from servers monitored by this EP.	`EventProcessorService.xml`

Setting	Default	Description	Location
		Note when the number of OSA events processed matches `eventsToProcessAtOnce` no wait is performed between checks for new OSA events to process. This is referred to as catchup mode. This property is only used when OSA event processing is done synchronously, which is controlled by the `MULTI_THREAD_OSA_EVENT_PROCESSING` property.	