QAD Adaptive Tuning Guide

Version 12.1 November 2025

QAD Adaptive Tuning Guide

Introduction

This document is intended to serve as a technical tuning guide for QAD Adaptive Production Servers. It is assumed that appropriate sized hardware has already been selected and commissioned. For further information on sizing QAD Adaptive systems, contact QAD Services.

Computer hardware is rapidly evolving and all QAD customers have different usage patterns, so the information in this document is of a general nature. All customer deployments are unique and some of the tuning recommendations in this document may not be appropriate for 100% of the customer base.

QAD Adaptive contains new technologies and components of the software are internally different from the previous versions of QAD Applications.

It is also important to remember that the QAD Adaptive tuning parameters are constantly being tested and analyzed to determine the optimal deployment options.

The tuning guide where applicable identifies yab configuration options to set the optimal values in blue.

Essential Requirements - Performance, Scalability, Stability

The items listed in this section are considered impactful enough to be considered a requirement or pre-requisite for an optimally tuned, scalable and stable system. This means ALL QAD systems should implement these settings.

The items are included based on direct experience with production systems, or as a result of recent new research. Each setting or procedure is described fully in its own section of the guide.

The following table summarizes the requirements and provides links to the relevant section in the guide.

Category	Area & Impacts / Rationale
-lruskips and -omsize Tuning	Not having these parameters tuned can cause global database performance issues and full business impairments
Ongoing OpenEdge Index maintenance	A lack of ongoing index maintenance (at least IDXCOMPACTS at 85%) causes query speed reductions and excessive disk space allocations for indexes. This negatively impacts global application performance to potentially high levels
Java metaspace tuning	Avoids tomcat outages due to metaspace triggered continuous GC (Applicable to all Java8+)
Nifi Max Usage Percentage	Avoids background processing outages (Applicable to all nifi installs)

Note: A setting or procedure not listed in this section does not mean that setting or procedure is not important, or critically important, for the environment. The items in this section are a subset of settings or procedures that from experience, or from new research, can cause critical issues, or have a very large or dominant impact on the non-functional areas of an environment's performance, stability or scalability.

OpenEdge Database Tuning

The following information is for a “quick default tune” for any OpenEdge database. After monitoring system performance, the tuning should be tailored to suit the individual system. Some essential requirements are also included.

-lruskips and -omsize Tuning (REQUIREMENT)

This area is impactful enough to be considered a ‘requirement’ for a well tuned and stable system (see the other essential requirements).

IMPORTANT: This is critical tuning for any database with moderate to high reads for the database startup parameters -lruskips and –omsize. **Not having this tuning in place has caused critical customer issues**.

-omsize

- The Storage Object Cache is used to hold copies of information about the most frequently accessed objects so that they can be found quickly, with minimal overhead. The optimal value is the number of storage objects + 10%.
- The current number of objects can be determined by running the following code from a Procedure Editor:
    - select count(*) from _storageobject.

yab configuration omsize

```
# Database omsize tuning

dbserver._base.storageobjectsize=1024

dbserver.extension.storageobjectsize=9100

dbserver.qaddb.storageobjectsize=10240

dbserver.qadeam.storageobjectsize=2800

dbserver.qadmodule.storageobjectsize=10240

dbserver.tmsdb.storageobjectsize=3500
```

-lruskips

- This parameter provides the ability to ease contention on the Buffer Pool Least Recently Used latch by altering the LRU replacement policy in the Primary (LRU).

yab configuration lruskips

```
#Database lruskips

dbserver._base.lruskips=100
```

OpenEdge Database Structure

- RM block size = 8k
- BI block size = 16k
- BI cluster size = 16Mb
- AI block size = 16k
- All data should be in storage area FIXED extents. If, after the conversion, any of the variable extents have grown, use prostrct add to refix these extents
- After-Image Sizing
    - Define eight (8) after-image extents, with the last one being a variable length.
    - The default size should be 5000 kB, but always use QAD Monitoring to help size more accurately.
    - Ensure the AI Archiver policy is to archive each extent after ten (10) minutes.
    - Set -aibufs to at least 1.5 times the number of -bibufs.
    - Use QAD Monitoring to determine whether -aibufs should be increased further.
    - Continuously roll forward the archived AI extents into a warm spare database.
    - Ensure AI block size = BI block size for all databases.
    - Ensure all production databases have After Image enabled.

OpenEdge Variable Extents

On existing databases, you should first consider the storage area's high-water mark. The high-water mark is established by the number of formatted blocks in an area. In an area with many empty (unformatted) blocks, data is allocated to the empty blocks before the system extends the last extent of the area. The goal is to never use the variable extent but to have it there if necessary.

The following is a small 4GL program that will give you the Area Name, Total Blocks in that area, and the High Watermark for that area:

```
FOR EACH _AreaStatus:

DISPLAY _AreaStatus-AreaName LABEL "Area Name" _AreaStatus-lastextent format “X(50)” LABEL "Last Extent" _AreaStatus-TotBlocks LABEL "Total Blocks" _AreaStatus-Hiwater LABEL "High Watermark" _AreaStatus-TotBlocks - _AreaStatus-Hiwater LABEL "Free" (SUM) .

END.
```

If the Last Extent being written to is a variable extent, then it is time to restructure the database by adding additional fixed extents for that storage area.

OpenEdge Database Fragmentation

A fragmented database can affect performance. Fragmentation can be checked with the following command:

```
$DLC/bin/proutil dbname -C dbanalys > index.txt
```

Look at the scatter factor. If > 4 to 4.5 on large tables, then schedule a dump/reload

			-Record Size (B)-			---Fragments---		Scatter
Table	Records	Size	Min	Max	Mean	Count	Factor	Factor
PUB.ih_hist	80279	32.4M	352	518	422	95599	1.2	6.6
PUB.iph_hist	273675	42.3M	140	179	162	273675	1.0	1.7
PUB.nrh_hist	2880660	256.2M	91	130	93	2880660	1.0	1.0
PUB.grig_mtx	19778793	1.4G	61	90	77	19778793	1.0	1.0
PUB.Posting	3357465	711.7M	178	238	222	3357465	1.0	1.5
PUB.PostingLine	7269638	1.2G	141	204	178	7269638	1.0	1.2

Database Startup Parameters

Ensure –B (**dbserver.INSTANCE.blocksindatabasebuffers**) is high enough for 99% buffer hit rate but not so high to cause system memory paging. As a starting point, set to 15% of the physical size of the database.

Other default parameters - please apply to main databases in use as needed

- -L 100000 –c 350 –n [usercount] –spin 50000 -aibufs 600 –bibufs 600 –Mf 6 –semsets 20 –tablerangesize 4096 – indexrangesize 1024 –T [fastest filesystem available] –lruskips 100 –omsize [Current object size + 10%]
- NOTE: Very large transactions may require a large -L parameter
- Use the AIW, BIW, Page Writers

yab configuration database

 # Database Tuning

dbserver._base.locktableentries=100000

dbserver._base.otherArgs=-c 350

dbserver._base.maxusers=[usercount]

dbserver._base.spinlockretries=50000

dbserver._base.tablerangesize=2048

dbserver._base.indexrangesize=1024

dbserver._base.afterimagebuffers=600

dbserver._base.beforeimagebuffers=600

dbserver._base.beforeimagedelaywrites=6

dbserver._base.semaphoresets=20

Database user connections

- Use a larger network buffer size (-Mm)

  yab configuration database Mm

  # Database -Mm

  dbserver.INSTANCE.fourgl.messagebuffersize

- Limit the maximum clients per DB server (-Ma)

  yab configuration database Ma

  # Database -Ma

  dbserver.INSTANCE.fourgl.maxclientsperserver

- Set the Maximum servers (-Mn)

  yab configuration database Mn

  # Database -Mn

  dbserver.INSTANCE.maxservers

- Set the minimum clients per server (-Mi)

  yab configuration database Mi

  # Database -Mi

  dbserver.INSTANCE.fourgl.minclientsperserver

OpenEdge APWS

- Start with 1
- Watch the value of buffers flushed at checkpoint during high usage periods
- If buffers flushed at checkpoint increases, add 1 more APW and repeat until the number of buffers flushed at checkpoint is zero or near zero

Ongoing Index Maintenance (REQUIREMENT)

This area is impactful enough to be considered an essential 'requirement' for an optimally tuned, scalable and stable system (see the other essential requirements).

If indexes are not maintained (compacted, rebuilt), queries become slower and the indexes occupy more space on disk than they need to. The query speed issue can impact global application performance. **There is no tuning parameter to fix this, only ongoing and focused maintenance of the indexes affected will work to return the indexes to their optimal (or near optimal) state**.

The operational procedures that can reduce or help minimize the above unavoidable index impacts are:

1. An online IDXCOMPACT at 85% run online, can restore indexes to an optimal state.
2. An offline index rebuild (IDXBUILD) restores the index to full optimal condition, but does require downtime.

The recommended requirement is as follows:

RECOMMENDED OPERATIONAL REQUIREMENT
INDEX MAINTENANCE
Create a batch job, run once per week, to apply online IDXCOMPACT at 85% to all indexes that breach the defined threshold ● Set a threshold for indexes to be compacted ● Specifically: ○ If index is <= 70 %utilization, consider it to be compacted ○ If index has an index factor >=1.7, consider it to be compacted ● Carry out an IDXCOMPACT at 85% for the target indexes that breach the threshold Notes: 1. As with any maintenance activity, the jobs should be tested and verified to run well. Older OpenEdge releases did have some IDXCOMPACT bugs, so a review of the safety of the utility should be reviewed in the context of the OpenEdge version running at the customer. 2. Offline IDXCOMPACTs or IDX REBUILDS can be considered, if that is an option. Offline rebuilds will make the most positive improvements.

For extra detail on this requirement see Appendix C: Requirements for an optimally tuned, scalable and stable system

Cassandra Database Tuning

Cassandra User Count 1-49

The following values should be set in the Cassandra jvm.options:

- -XX:+UseG1GC
- -XX:MaxGCPauseMillis=500

yab configuration cassandra 1-49 users

```
# Cassandra Database Tuning

cassandra.default.node.jvmoptions.useg1gc=/^#-XX:\+UseG1GC/-XX:+UseG1GC

cassandra.default.node.jvmoptions.minheap=/^#-Xms.*$/-Xms512m

cassandra.default.node.jvmoptions.maxheap=/^#-Xmx.*$/-Xmx512m
```

Cassandra User Count 50+

The following values should be set in the Cassandra jvm.options:

- -Xms1G
- -Xmx1G
- -XX:+UseG1GC
- -XX:MaxGCPauseMillis=500

yab configuration cassandra 50+ users

```
# Cassandra Database Tuning

cassandra.default.node.jvmoptions.useg1gc=/^#-XX:\+UseG1GC/-XX:+UseG1GC

cassandra.default.node.jvmoptions.minheap=/^#-Xms.*$/-Xms1G

cassandra.default.node.jvmoptions.maxheap=/^#-Xmx.*$/-Xmx1G
```

Application Server Tuning - 'PASOE' OpenEdge 12

In OpenEdge 12, Progress have deprecated the classic appserver and replaced it with the Progress Application Server for OpenEdge (PASOE). PASOE is an entirely different architecture than the Classic Appserver. Further details on the PASOE Architecture are outlined in Appendix B.

This section refers to tuning for the PASOE Appserver.

OE12 - ABL Sessions & Multi-Session Agents (Min / Max / Initial / Clients)

The following table lists the key tuning point from a QAD tuning perspective:

PASOE parameter (Yab key)	Comment / Labeling for this document
pas.INSTANCE.openedge.agent	Multi-Session Agent YAB definition key
pas.INSTANCE.openedge.session-mgr	Session Manager YAB definition key
pas.INSTANCE.openedge.session-mgr.minAgents	minAgent
pas.INSTANCE.openedge.session-mgr.maxAgents	maxAgent
pas.INSTANCE.openedge.session-mgr.numInitialAgents	initAgent
pas.INSTANCE.openedge.agent.numInitialSessions	initABLSess
pas.INSTANCE.openedge.agent.minAvailableABLSessions	minAvailABLSess
pas.INSTANCE.openedge.session-mgr.maxABLSessionsPerAgent	maxABLAgentSess
pas.INSTANCE.openedge.session-mgr.maxConnectionsPerAgent	maxAgentConn
pas._base.openedge.session-mgr.minAgents	minMSAgents

Table: A summary of the primarily relevant PASOE parameters related to QAD tuning - these settings are related to the Multi-Session Agent configuration and to the pool of ABL sessions. The full yab keys are listed together with a shorthand label, the label is only a shorthand for use in this document.

The following table shows the recommended tuning parameters:

PASOE ABL Sessions & Multi Session Agent tuning				
Config Setting [Label]	Small (1-49)	Medium (50-199)	Large (200-499)	A+ (500+)
	QRA / FIN / MFG / Other Appservers & Webspeed			
initAgent	1/3/1/1	1/3/1/1	1/3/1/1	1/3/1/1
minAgent	1/3/1/1	1/3/1/1	1/3/1/1	1/3/1/1
maxAgent	3/3/3/3	3/3/3/3	3/3/3/3	3/3/3/3
initABLSess	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1
minAvailABLSess	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1
maxABLAgentSess	10/10/10/5	15/15/15/8	15/20/20/8	15/25/25/8
maxAgentConn	10/10/10/5	15/15/15/8	15/20/20/8	15/25/25/8
	Max concurrent ABL Sessions (i.e. client requests)			
	30/30/30/15	45/45/45/24	45/60/60/24	45/75/75/24

- **Note 1:** These settings are based on sizing to have 3 multi session agents, max with 1 minAgent, and on distributing ABL Sessions appropriately across these agents. This provides for a much more efficient use of server and PASOE RAM, while preserving performance and scalability under load.
- **Note 2**: maxAgentConn should always match maxABLAgentSess
- Note 3: “Max concurrent ABL Sessions” possible for PASOE = maxAgent x maxABLAgentSess, and is analogous, in terms of maximum concurrent simultaneous request processing, to the maxSrvrInstance setting for a classic appserver
- **Note 4:** A perceptive reader will notice that the maximum levels for PASOE ABLSessions are lower than the classic appserver “maxSrvrInstance” levels. Previously, for the classic appserver, we have sized the maxSrvrInstance to 100/100/50 (i.e. 100 for the QRA, MFG and FIN appservers, 100 for other appservers, and 50 for webspeed). However, the “maxSrvrInstance” value is used as an “overflow”/”release valve” mechanism where by if there is a large increase in load that requires more appserver agents, then more can be created, up to the maxSrvrInstance level. Normally the minSrvrInstance levels should be more

than sufficient to handle the load for the given customer size. We have found that while providing a large “overflow”/release value is good in theory, the large amount of extra RAM required from so man extra processes can cause global issue due to server-wide memory starvation, and the increased concurrent demand on the number of concurrent CPU cores can cause CPU subsystem overload (100 CPU cores is not common). Ultimately, the release valve, if fully released, can cause larger global issues that instead limiting the release to a more realistic and manageable level. Based on this we have reduced the maximum concurrently levels in PASOE.

So, taking an example, here is what the yab configuration would look like for a large system of 350 users:

yab configuration 350 user PASOE example

 # Base webspeed MSAs

pas._base-classic-ws.openedge.session-mgr.minAgents=1

pas._base-classic-ws.openedge.session-mgr.maxAgents=3

pas._base-classic-ws.openedge.session-mgr.numInitialAgents=1

 # Base PASOE MSAs

pas._base.openedge.session-mgr.minAgents=1

pas._base.openedge.session-mgr.maxAgents=3

pas._base.openedge.session-mgr.numInitialAgents=1

 # FIN PASOE MSAs

pas._base.openedge.session-mgr.minAgents=3

pas._base.openedge.session-mgr.maxAgents=3

pas._base.openedge.session-mgr.numInitialAgents=3

 # Base Webspeed ABL Sessions per Agent

pas._base-classic-ws.openedge.agent.minAvailableABLSessions=1

pas._base-classic-ws.openedge.session-mgr.maxABLSessionsPerAgent=8

pas._base-classic-ws.openedge.agent.numInitialSessions=1

 # Base PASOE appserver ABL Sessions per Agent

pas._base.openedge.agent.minAvailableABLSessions=1

pas._base.openedge.session-mgr.maxABLSessionsPerAgent=8

pas._base.openedge.agent.numInitialSessions=1

 # QRA-, FIN- and MFG-specific PASOE appserver settings

pas.qra.openedge.session-mgr.maxABLSessionsPerAgent=15

pas.qra.openedge.session-mgr.maxConnectionsPerAgent=15

pas.fin.openedge.session-mgr.maxABLSessionsPerAgent=20

pas.fin.openedge.session-mgr.maxConnectionsPerAgent=20

pas.mfg.openedge.session-mgr.maxABLSessionsPerAgent=20

pas.mfg.openedge.session-mgr.maxConnectionsPerAgent=20

Additional Parameters - All Configurations

yab configuration collect metrics

 # Add full collectMetrics diagnostics for QRA / MFG / FIN

pas.qra.openedge.appserver=AppServer

pas.qra.openedge.appserver.collectMetrics=3

pas.mfg.openedge.appserver=AppServer

pas.mfg.openedge.appserver.collectMetrics=3

pas.fin.openedge.appserver=AppServer

pas.fin.openedge.appserver.collectMetrics=3

OE12 - Non-PROD Tuning Guidelines

For non-production systems certain PASOE tuning can be reduced, which helps optimize sizing and reduce infrastructure demands, while delivering standard performance. By this we mean systems that have a concurrent user count where that user base is developing or testing, rather than running production level business operations, for example development and test systems. The following are the recommendations for non-Production systems.

PASOE ABL Sessions & Multi Session Agent tuning NON-PROD Systems		
Config Setting [Label]	Non-Prod Tuning	Medium - Large (50+)
	QRA, FIN & MFG Appservers / Other Appservers / Webspeed	
initAgent	1/1/1	1/1/1
minAgent	1/1/1	1/1/1
maxAgent	**1/1/1**	**1/1/1**
initABLSess	1/1/1	1/1/1
minAvailABLSess	1/1/1	1/1/1
maxABLAgentSess	10/5/5	15/8/8
maxAgentConn	10/5/5	15/8/8
	Max concurrent ABL Sessions (i.e. client requests)	
	10/5/5	**15/8/8**

Table: The assumption for non-PROD systems is that they are not used to run a production level load, but are instead used for development and testing. As such, they do not require the same sizing levels, or require the same “buffer”/”release value” levels for peak production load. Reductions from the production tuning levels are highlighted in **red**.

- **Note 1**: The recommendation is to have one Multi Session Agent configured.
- **Note 2**: If having one Multi Session Agent configured is causing a restriction in processing, then add one more Multi Session Agent to the appserver, rather than increasing the number of ABL Sessions in the original Multi Session Agent.

PAS Agent Server Startup Parameters (session-mgr.agentStartupParam)

These recommendations apply to the QRA (pas-qra), Financials (pas-fin), and UI (pas-mfg) PAS session-mgr.agentStartupParam with the primary differences between each appserver highlighted in bold.

For the QRA appservers

> **–mmax 49152** -c 30 –D 1000 –s 160 -nb 200 –inp 32000 -tok 20000 –TB 31 –TM 32 –Bt 20000 -tmpbsize 4 -q -T [Fastest available disk] -rereadnolock -noshvarfix -h 25 -noduptidxerror -errorstack -zNoQryExpErrors

For the FIN appservers

> **–mmax 8192** -c 30 –D 1000 –s 160 -nb 200 –inp 32000 -tok 20000 –TB 31 –TM 32 –Bt 20000 -tmpbsize 4 -T [Fastest available disk] -rereadnolock -noshvarfix -h25 -noduptidxerror

For the MFG and other appservers

> **–mmax 20408** -c 30 –D 1000 –s 160 -nb 200 –inp 32000 -tok 20000 –TB 31 –TM 32 –Bt 20000 -tmpbsize 4 -T [Fastest available disk] -rereadnolock -noshvarfix -h25 -noduptidxerror

For the QXtend appservers

> **–mmax 8192** -c 30 –D 1000 –s 160 -nb 200 –inp 32000 -tok 20000 –TB 31 –TM 32 –Bt 20000 -tmpbsize 4 -T [Fastest available disk] -rereadnolock -noshvarfix -h25 -noduptidxerror

NOTE: -q should only be used on Production Servers

> What the -q parameter does:
> Normally, when an r-code file is loaded, the AVM will always search the PROPATH looking for a a matching r-code file (a file with the same name as the procedure, but with a .r extension). If a matching r-code file is found it checks if the version in memory is still the same as the version on disk, and re-loads the r-code if it did change. This check ensures the client always uses the most recent r-code file.
>
> The Quick Request startup parameter (-q) changes this behavior, and effectively disables this check. If the r-code still resides in memory, the AVM will re-use the copy in memory without checking the copy on disk. This reduces the overhead involved in running the same procedures multiple times, therefore the -q parameter can be used to improve performance in environments that run on a stable set of code, such as production and stress-testing environments.

yab configuration application server session-mgr.agentStartupParam

```
# PAS Server Tuning

# Common Tuning

pas-common.tuning=-c 30 –D 1000 –s 160 -nb 200 –inp 32000 -tok 20000 –TB 31 –TM 32
–Bt 20000 -tmpbsize 4 -q -T ${tmp} -cpinternal ${db._base.codepage} -cpstream
${db._base.codepage} -cpcoll ${db._base.collation} -cpcase
${dbserver._base.casetablename} -rereadnolock -noshvarfix -h25 -noduptidxerror

#QRA Tuning

pas.qra.openedge.session-mgr.agentStartupParam=-pf ${bootstrap.default.parameterfile}
${pas-common.tuning} -mmax 49152 -q

#FIN Tuning

pas.fin.openedge.session-mgr.agentStartupParam=-pf ${${fin.bootstrap}.parameterfile}
${pas-common.tuning}  -mmax 8192

#MFG Tuning

pas.mfg.openedge.session-mgr.agentStartupParam=-pf ${bootstrap.default.parameterfile}
${pas-common.tuning} -mmax 20408

#QXtend Tuning

pas.qxtnative.openedge.session-mgr.agentStartupParam=-pf
${bootstrap.default.parameterfile} ${pas-common.tuning} -mmax 20408

#Other AppServer Tuning

pas.INSTANCE.openedge.session-mgr.agentStartupParam=-pf
${bootstrap.default.parameterfile} ${pas-common.tuning} -mmax 20408
```

PAS Server Trimming

OpenEdge 12 Trimming - Not YAB

Trimming in OE12 should NOT use yab but use the custom OE12 script until yab has been adjusted to use terminateFreeABLSession(s).

The custom trimming script is available in google drive: trim_PASOE.sh using the following crontab entry:

```
*/10 * * * * /qond/apps/mfgpro/scripts/trim_PASOE.sh >>
/qond/apps/mfgpro/scripts/logs/trim_PASOE_$(date +\%Y\%m\%d).log
```

Further details on PASOE ABL Session lifecycle management is outlined in Appendix B.

Connection Manager Tuning

The pf file used by the connection manager should have the following tuning applied:

```
-T [Fastest available disk] -d mdy -yy 1950 –s 32768 –mmax 20408 –inp 32000 -rereadnolock
-c 30 –D 1000 –Bt 20000  -nb 200 -h 25 –inp 32000 -tok 20000 –TB 31 –TM 32 -tmpbsize 4
```

yab configuration progres tuning

```
# Set Default Production Tuning

progress.startup.params=-T ${tmp} -d ${progress.dateformat} -yy ${progress.centuryoffset}
–s 32768 –mmax 20408 –inp 32000 -rereadnolock -c 30 –D 1000 –Bt 20000  -nb 200 -h
25 –inp 32000 -tok 20000 –TB 31 –TM 32 -tmpbsize 4
```

Connection Manager Connections

- MaxConnections: 1000

  yab configuration connection manager max connections

  ```
  # Connection Manager Max Connections

  netui.connmgr.maxconnections=1000
  ```

- MinConnections: 4 for Small and 20 for other customers

yab configuration connection manager min connections

 # Connection Manager Min Connections

netui.connmgr.minconnections=4 or netui.connmgr.minconnections=20

Tomcat / Java Tuning

The performance tuning aspects of Java are dependent on the number of concurrent QAD users. Some tradeoffs to absolute responsiveness must be made to ensure high user count stability. In this section, the recommended tuning parameters are based on the expected number of concurrent users.

Support for High User Counts (NETUI and WEBUI Tomcats)

By default, the maximum threads (connection manager threads) is 150.

To allow for more than 150 threads, the $TOMCAT/conf/server.xml must be modified to raise maxThreads:

<Connector port="8080" maxHttpHeaderSize="8192" **maxThreads="1000"** minSpareThreads="25" maxSpareThreads="75" enableLookups="false" redirectPort="8443" acceptCount="100" connectionTimeout="20000" disableUploadTimeout="true" />

yab configuration tomcat threads

 # Set Maximum Threads

tomcat.default.maxthreads=1000

tomcat.webui.maxthreads=1000

JVM metaspace tuning (REQUIREMENT)

- It is mandatory to have the **metaspace configuration** as per below to prevent tomcat outages:
  - -XX:MetaspaceSize=1024M -XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100 -XX:MinMetaspaceFreeRatio=0

JVM WEBUI Tuning

Please set the following property to increase the tomcat-webui cache size:

tomcat.webui.context.resources.cacheMaxSize=102400

yab configuration webui 1-199 users

tomcat.webui.startupopts=-server -Xms4g -Xmx4g -XX:MetaspaceSize=1024M -XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100 -XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch -Xlog:gc*:${appdir.logs}/tomcat_webui_gc.log:time,level,tags -Xss512k -XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp} -Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/sun.util.calendar=ALL-UNNAMED

yab configuration webui 200+users

tomcat.webui.startupopts=-server -Xms8g -Xmx8g -XX:MetaspaceSize=1024M
-XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100
-XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200
-XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch
-Xlog:gc*:${appdir.logs}/tomcat_webui_gc.log:time,level,tags -Xss512k
-XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp}
-Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/sun.util.calendar=ALL-UNNAMED
--add-opens=java.xml/com.sun.org.apache.xerces.internal.dom=ALL-UNNAMED
--add-opens=java.xml/com.sun.org.apache.xerces.internal.jaxp=ALL-UNNAMED
--add-opens=java.xml/com.sun.org.apache.xerces.internal.xni=ALL-UNNAMED

JVM NETUI Tuning

yab configuration netui 1-199 Users

tomcat.default.startupopts=-server -Xms1g -Xmx1g -XX:MetaspaceSize=1024M
-XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100
-XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200
-XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch
-Xlog:gc*:${appdir.logs}/tomcat_netui_gc.log:time,level,tags -Xss512k
-XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp}
-Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/sun.util.calendar=ALL-UNNAMED

yab configuration netui 200-499 Users

tomcat.default.startupopts=-server -Xms2g -Xmx2g -XX:MetaspaceSize=1024M
-XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100
-XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200
-XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch
-Xlog:gc*:${appdir.logs}/tomcat_netui_gc.log:time,level,tags -Xss512k
-XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp}
-Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED

--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/sun.util.calendar=ALL-UNNAMED

yab configuration netui 500+ Users

tomcat.default.startupopts=-server -Xms4g -Xmx4g -XX:MetaspaceSize=1024M
-XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100
-XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200
-XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch
-Xlog:gc*:${appdir.logs}/tomcat_netui_gc.log:time,level,tags -Xss512k
-XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp}
-Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/sun.util.calendar=ALL-UNNAMED

Note1 : For each additional 50 users, add 100 MB to the –Xms -Xmx

JVM Event Service Tuning

- Event service tuning should be adjusted based on expected activity tracking

yab configuration event service

```
tomcat.eventservice.startupopts=-server -Xms1g -Xmx1g -XX:MetaspaceSize=1024M
-XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100
-XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200
-XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch
-Xlog:gc*:${appdir.logs}/tomcat_event_servicei_gc.log:time,level,tags -Xss512k
-XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp}
-Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true
```

JVM QXtend Tuning

- QXtend tuning should be adjusted based on expected activity / transaction throughput

yab configuration qxtend

```
tomcat.qxtend.startupopts=-server -Xms1g -Xmx1g -XX:MetaspaceSize=1024M
-XX:MaxMetaspaceSize=1024M -XX:MaxMetaspaceFreeRatio=100
-XX:MinMetaspaceFreeRatio=0 -XX:+UseG1GC -XX:MaxGCPauseMillis=200
-XX:InitiatingHeapOccupancyPercent=75 -XX:+AlwaysPreTouch
-Xlog:gc*:${appdir.logs}/tomcat_qxtend_gc.log:time,level,tags -Xss512k
-XX:-HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${tmp}
-Djava.net.preferIPv4Stack=true -Dfile.encoding=utf-8 -Djava.awt.headless=true
```

JVM Kafka Tuning

yab configuration kafka

```
# Kafka Tuning

kafka.default.env.heapopts.value=-Xmx256M -Xms256M
```

JVM Elastic Tuning

yab configuration elastic

```
# elastic tuning

elasticsearch.default.jvm.maxheap=/^-Xmx.*$/-Xmx256m
```

elasticsearch.default.jvm.minheap=/^-Xms.*$/-Xms256m

JVM Nifi Tuning

yab configuration nifi user count 1-49

 # Nifi Tuning

nifi._base.bootstrap.java.arg.2=-Xms256M

nifi._base.bootstrap.java.arg.3=-Xmx512M

yab configuration nifi user count 50+

 # Nifi Tuning

nifi._base.bootstrap.java.arg.2=-Xms1g

nifi._base.bootstrap.java.arg.3=-Xmx1g

Nifi Background Processing Outage Prevention (REQUIREMENT)

- Set nifi.content.repository.archive.max.usage.percentage=90% to avoid background processing from stopping when the main disk device becomes 52% full

yab configuration nifi outage prevention

 # Nifi Tuning

nifi.default.application.nifi.content.repository.archive.max.usage.percentage=90%

JVM Zookeeper Tuning

yab configuration zookeeper

 # Zookeeper Tuning

zookeeper._base.env.heapopts.value=-Xmx128M -Xms128M

zookeeper.default.env.heapopts.value=-Xmx128M -Xms128M

Application Configuration

server.xml

fin.serverxml.swap-limit=20

fin.serverxml.reports-batch-size=5000

<swaplimit>

default = 20

</swaplimit>

<database>

Ensure that all database connections are shared memory only

</database>

<batchsize>

default = 5000 initial recommendation = 1000 (check with Application Consultants) </batchsize>

<state>

Initial recommendation = leave value blank (state will be stored in the database)

</state>

Event Service Tuning

- eventservice.publisher.entityfetcherconcurrency should have the following settings to control the number of Activity Feed fetch threads:

  yab configuration entityfetcherconcurrency (< 100 users)

  # Event Service Concurrency

  eventservice.publisher.entityfetcherconcurrency=2

  yab configuration entityfetcherconcurrency (100-500 users)

  # Event Service Concurrency

  eventservice.publisher.entityfetcherconcurrency=4

  yab configuration entityfetcherconcurrency (> 500 users)

  # Event Service Concurrency

  eventservice.publisher.entityfetcherconcurrency=6

- dataChangePublisher.messageGenerator.concurrency should have the following settings to control the number of publisher threads

  yab configuration dataChangePublisher (< 100 users)

  # Data Change Concurrency

  eventservice.publisher.messagegeneratorconcurrency=1

  yab configuration dataChangePublisher (100-500 users)

  # Data Change Concurrency

  eventservice.publisher.messagegeneratorconcurrency=3

  yab configuration dataChangePublisher (> 500 users)

  # Data Change Concurrency

  eventservice.publisher.messagegeneratorconcurrency=5

- jms queue configuration (Assuming a 1GB Heap). The general rule is that global-max-size should be 50% the size of the heap; max-size-bytes 10% the size of the heap; and page-size-bytes 1% the size of the heap. If you increase the heap, the previous values should also be increased.

  yab configuration jms queue

  # JMS Queue Tuning

  eventservice.jms.global-max-size=524288000

  eventservice.jms.max-size-bytes=104857600

  eventservice.jms.page-size-bytes=10485760

Query Service Tuning Parameters

The following settings should be considered when setting up the Query Service.

The following app settings are located in qad-analytics-core.properties but changes to defaults should only be made in build/config/configuration.properties:

yab configuration scheduledRefresh

 # Analytics KPIs Tuning

qad-analytics-core.cache.kpis-browse.scheduledRefresh.quartzJobCount=2

qad-analytics-core.cache.kpis-browse.scheduledRefresh.batchSize=100

- batchSize should be large enough for the majority of KPIs to be refreshed in a single run of a thread. For example, if there are 220 KPIs in the system and a Quartz Job Count of 2, then a setting of 100 should be adequate.
- The following formula should be used as a guideline: Batch Size = (Number of KPIs / Quartz Job Count)

The following app settings can be found in qad-qracore.properties but changes to defaults should be made in build/config/configuration.properties:

yab configuration batchSize

 # Cassandra Browse Tuning

qad-qracore.browseCassandraDataService.batchSize=1000

qad-qracore.browseCassandraDataService.pageSize=5000

qad-qracore.browseCassandraDataService.timeoutLimit=120

JDBC Connections and Hikari

The recommended default Hikari pool size is as follows:

yab configuration hikari pool

 # Hikari Pool Tuning

qad-qracore.hikari.centraldb.maxpoolsize=10

qad-qracore.hikari.maxpoolsize=5

This pool is primarily used for the following:

- Tomcat startup for loading permissions
- Tomcat startup to refresh the KPIs for Action Centers
- Platform to create / update / delete for deployed applications

It is recommended to increase the max pool size by 1 for each 10 users of a platform customization. Monitoring of this pool is recommended.

Appendix A – Reference Material

- Iruskips and omsize critical bottleneck
- What does -q do?
- QAD Adaptive 2024 SP1 ERP Tuning Guide - November 2025
- tomcat/java – Approach to stability and tuning
- Tomcat outage analysis and prevention
- Logi Java Server Configurations

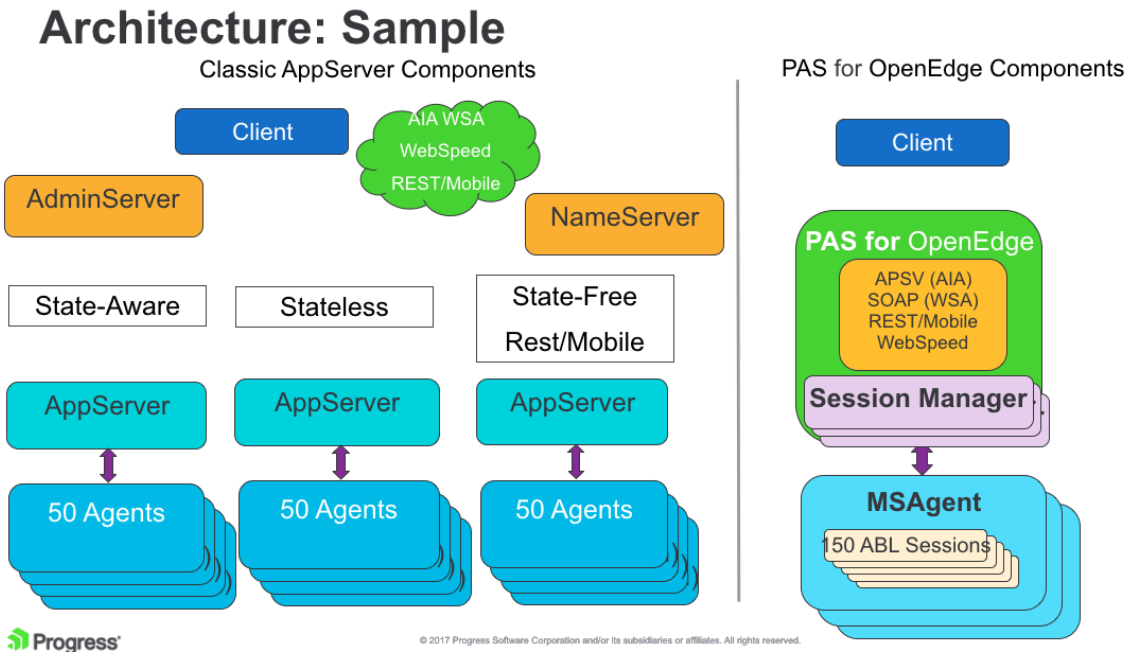
Appendix B – PASOE

Introduction

In OpenEdge 12, Progress have deprecated the classic appserver and replaced it with the Progress Application Server for OpenEdge (PASOE). PASOE is an entirely different architecture than the Classic Appserver, as it is a customized tomcat based application server using HTTP sessions, linked with a separate multi-threaded “Agent” process. As such, PASOE comes with a large array of new tuning options and new tuning parameters related to the new architecture.

PASOE architecture

The following is a high level schematic of the PASOE architecture.



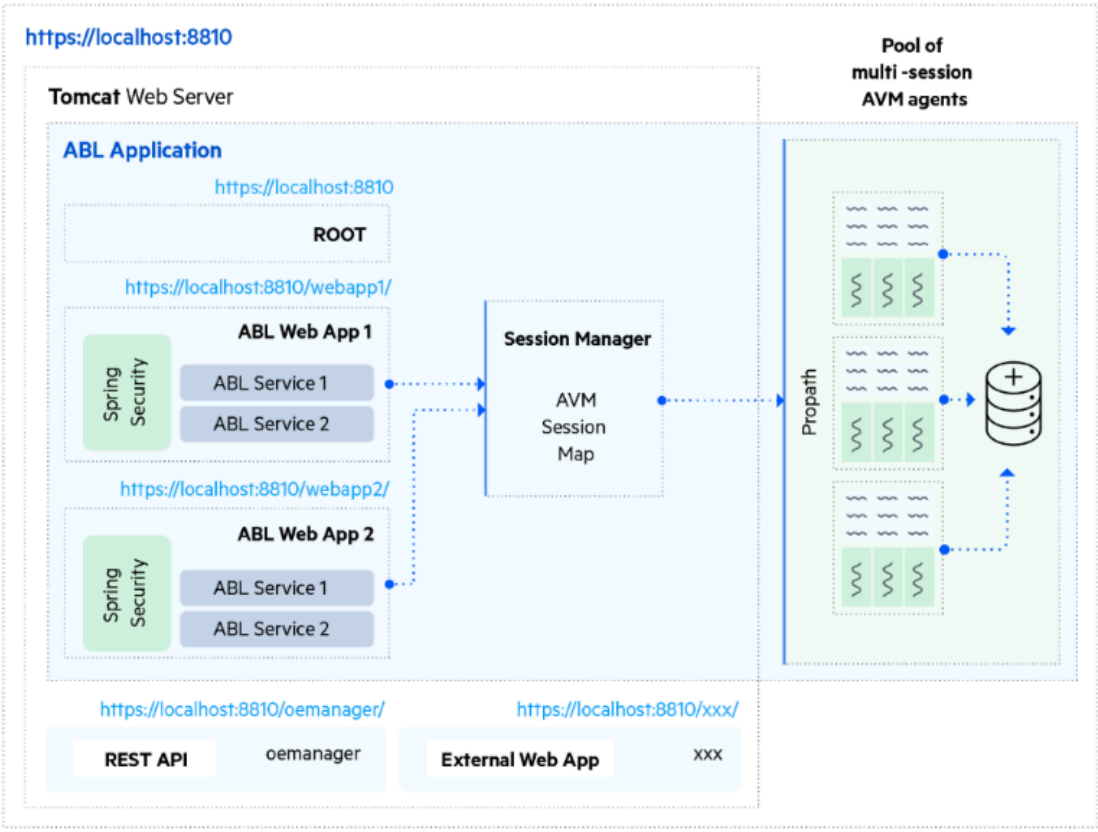


Figure: The main components of each PASOE instance (see the Progress documentation).

- The Progress ABL code is run within the ABL Sessions ("AVM agents") of the Multi Session Agent
- The (customized by Progress) tomcat web server on the left handles client requests using a HTTP session model.
- All previous protocols like REST, SOAP, AIA / WSA are now all handled in one place within the customized tomcat web server, removing the need for separate AIA adaptors for example to run in separate java instances.
- The modified tomcat web server runs as a Progress java process and the "multi-session agent" will appear as a separate Progress "_mproapsv" process on the server.
- Each PASOE appserver can have one or more multi session agents (MSA)
- Each MSA has a pool of ABL sessions.
- Each ABL session runs one single client request at a time
- ABL sessions run as OS thread within the ABL pool, and multiple threads can run in parallel

YAB settings and for PASOE

The following are part of the changes from the deprecation of the Classic appserver, and the move to the new PASOE:

- Classic appserver management commands are removed:
  - asbman, nsman, wtbman, wsaman
- All protocol connections are now handled by the PASOE tomcat web application natively within the web app as "transports". This removes the need for having separate "adapters" in a separate web server.

These changes mean that for yab, all PASOE commands now use the following prefix:

- pas (instead of the previous "appserver" prefix for the classic appserver)

Try the following for a listing of YAB commands related to PASOE:

- yab help pas- # all PASOE commands
- yab help pas-qra # all PASOE commands related to the QRA appserver

ABL Sessions & Multi-Session Agents (Min / Max / Initial / Clients)

ABL-Sessions running within the PASOE Multi-Session Agent (MSA) now run the Progress business logic.

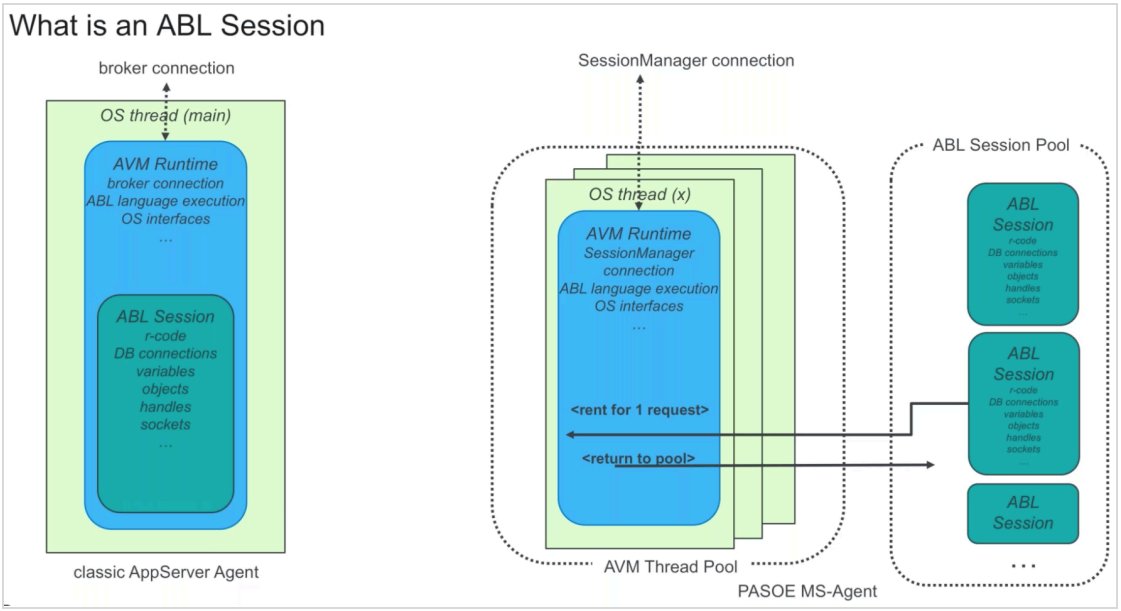


Figure: Each PASOE appserver can have one or more Multi-Session agents configured, where each MSA contains a (configurable) pool of ABL Sessions.

PASOE ABL Session lifecycle management (“Trimming”)

PASOE ABL Session lifecycle management is different than the classic appserver.

- PASOE can consume significantly more RAM than the classic appserver if this lifecycle is not properly managed through appropriate trimming.
- Trimming in PASOE is also currently more costly (in terms of CPU demand and in terms of duration for trimming) than classic appserver trimming.
- PASOE trimming becomes an important tool to keep RAM sizing down, and to prevent excessive and unnecessary RAM growth
- **Deviation from the recommended trimming approach for PASOE may have negative consequences**

The recommended trimming approach for PASOE is as follows:

Recommended trimming approach for PASO	
Multi Session Agent (MSA) trimming	1. Always retain at “minAgent” MSA per PASOE appserver a. minAgent by default should be 1, i.e. never trimm ALL MSAs b. E.g. minAgent=1, # active MSAs= 2, then we have 1 MSA potentially available to trimm 2. Trimm the MSAs if: a. The MSA has been inactive for >= 10 mins,
ABL Session trimming	Trim the ABL Session with largest memory from each MSA every 10 minutes (i.e. trimm one ABL session per MSA every 10 minutes)

Example recommended crontab yab command entry for ABL Session trimming:

- trim largest ABL Session (by memory footprint) once every 10 minutes for each appserver:

```
*/10 * * * * [ -d /dr01/qadapps/systest/servers/pas-qra ] &&  [ -d
/dr01/qadapps/yab/yab ] && /dr01/qadapps/yab/yab
-a:/dr01/qadapps/systest pas-trim -r:false -trim-max:1 -trim-with-min:true
-trim-by-size:true -failonerror:false -log-level:off > /dev/null 2>&1
```

Appendix C - Requirements for an optimally tuned, scalable and stable system

This appendix provides some background and some extra detail behind certain items listed in the essential requirements section. In this appendix:

- OpenEdge Index Maintenance
- OpenEdge Appserver - QueueLimit
- OpenEdge Appserver - maxSrvrInstances
- Java metaspace

OpenEdge Index Maintenance

This section provides some extra details on the impacts of NOT maintaining OpenEdge indexes on a regular basis. For the recommendation and requirements in this area see the main OpenEdge Index Maintenance section.

Impacts

Over time, if an index is used but not 'maintained' then that index can become underutilized (in the Progress sense of the word) and gets into a state that is far from optimal. By maintenance in this case, we mean applying regular or periodic IDXCOMPACTs, IDXBUILDs or D+Ls to the index.

Note: By 'utilization' here we mean the formal Progress metric of index '%utilization' as reported in a proutil DBANALYSIS report for the index.

For example:

- An index with 60 %utilization, means that it has approximately 40% of its allocated disk space empty.
- The database read performance of underutilized indexes is also negatively effected, anywhere from 10% slower to over 100% slower when compared to when that index has high utilization.

These index impacts are an unavoidable result of database activity (creation, deletion, update of records). As the database and index grows, these effects can worsen, effectively wasting valuable disk space and negatively impacting application performance (as the underlying index query speeds are reduced). The only way to reduce or minimize these impacts is via appropriate operational maintenance - next.

Notes

1. Neither the IDXCOMPACT online or the offline index rebuild process is capable of freeing up the on-disk "empty" space from an underutilized index. Once the empty space gets in there (through normal, unavoidable activity), it can not be recovered from disk. This is one reason why it's important to maintain indexes from the start, which will help minimize the build up of empty space in indexes through regular index maintenance procedures.
2. The reason for why disk space is not freed up is because, when using Type 2 storage areas, the type 2 clusters already dedicated to the underutilized index can not be deleted from the storage area (even if there are many empty index clusters after the compact), they have to be retained in that storage area and they can only be populated by that same index object and not by any other object (another table or index, or BLOB etc). This is a rule of the OpenEdge storage engine. Any empty index clusters will be available for future expansion of the index, but they are not deleted by the storage engine.

Java metaspace

Impact and Reference

- https://team.qad.com/display/QPT/Tomcat+Outages+2023#TomcatOutages2023-PrimaryCauseofoutages:MetaspaceTuning-GlobalCloudChangesRecommended

Appendix D - Revision History

Version 6.3

- Adjusted based on March-2021parameters

Version 6.2

- Migrated from Sept-2020 toMarch-2021
- Applied new template

Version 6.1

- Adjusted based on suggestions from Shane O'Riordan

Version 6.0

- Migrated from March-2020 to Sept-2020

Version 6.3

- March-2021
- Updated tuning for < 50 users
- Updated based on comments from djh@qad.com

Version 7.0

- Updated for Sept-2021

Minor changes

Version 8.0

- Updated for March-2022

Minor changes

- Version 9.0
- Updated for September-2022 to include OpenEdge12

Version 9.1

- Moved PASOE details to Appendix B
- Aligned Appserver Java Tuning
- Minor changes to formatting

Version 10.0

- "Requirements" section added
- To highlight areas having known impacts to global stability / global performance

Version 11.0

- Applied new template
- Added Yab setting added in relation to queueLimit for OE11
- Added requirement for Java metaspace tuning changes to avoid outages
- Added Java GC metric parameter adjustments to support Java17
- Added Nifi parameter adjustment to avoid background processing outages
- Some formatting changes
- Removed a small amount of content for tidy up purposes

Version 12.0

- Removed OpenEdge 11 references

- Reviewed content
- Resolved naming convention

Version 12.1

- Increased pas-mfg and pas-fin connection details based on customer issues
- Added tomcat.webui.context.resources.cacheMaxSize=102400 for large cache