



QAD Adaptive Tuning Guide

August 2026

Version 13.0



Contents

Introduction	5
Essential Requirements - Performance, Scalability, Stability	6
Change Control - Important General Principles	7
Validation Process before making any Memory Related Changes	7
Safety Considerations for Runtime Changes: Red-Amber-Green Classification	8
Making Tuning Changes	8
OS RAM Tuning dirty_bytes	9
OpenEdge Database Tuning	9
-lruskips and -omsize Tuning (REQUIREMENT)	9
-omsize	9
yab configuration omsize	10
-lruskips	10
yab configuration lruskips	10
OpenEdge Database Structure	10
OpenEdge Variable Extents	11
OpenEdge Database Fragmentation	11
Database Startup Parameters	12
yab configuration database	12
Database user connections	13
yab configuration database Mm	13
yab configuration database Ma	13
yab configuration database Mn	13
yab configuration database Mi	13
OpenEdge APWS	13
Ongoing Index Maintenance (REQUIREMENT)	14
Cassandra Database Tuning	15
Cassandra User Count 1-49	15
yab configuration cassandra 1-49 users	15
Cassandra User Count 50+	15
yab configuration cassandra 50+ users	15
PASOE OpenEdge 12 Tuning	16
PASOE OpenEdge 12 - PROD Tuning Guidelines	16
PAS Agent Server Startup Parameters (session-mgr.agentStartupParam)	18
PASOE OpenEdge 12 jemalloc	20

PASOE OpenEdge 12 - NON-PROD Tuning Guidelines	21
PASOE OpenEdge 12 Trimming	23
OpenEdge 12 Trimming - Not YAB	23
Specific QAD Application Tuning	24
MongoDB cacheSizeGB	24
PASOE jemalloc	25
NiFi Content Repository Archive Percentage	27
Connection Manager Tuning	28
yab configuration progres tuning	28
Connection Manager Connections	28
yab configuration connection manager max connections	28
yab configuration connection manager min connections	28
Tomcat / Java Tuning	29
Support for High User Counts (NETUI and WEBUI Tomcats)	29
yab configuration tomcat threads	29
JVM metaspace tuning (REQUIREMENT)	30
JVM WEBUI Tuning	30
yab configuration webui 1-199 users	30
yab configuration webui 200+users	31
JVM NETUI Tuning	31
yab configuration netui 1-199 Users	31
yab configuration netui 200-499 Users	32
yab configuration netui 500+ Users	32
JVM Event Service Tuning	33
yab configuration event service	33
JVM QXtend Tuning	33
yab configuration qxtend	33
JVM Kafka Tuning	33
yab configuration kafka	33
JVM Elastic Tuning	34
yab configuration elastic	34
JVM Nifi TuningXmx	34
yab configuration nifi user count 1-49	34
yab configuration nifi user count 50+	34
Nifi Background Processing Outage Prevention (REQUIREMENT)	34
JVM Zookeeper Tuning	35

yab configuration zookeeper	35
Application Configuration	36
server.xml	36
Event Service Tuning	37
yab configuration entityfetcherconcurrency (< 100 users)	37
yab configuration entityfetcherconcurrency (100-500 users)	37
yab configuration entityfetcherconcurrency (> 500 users)	37
yab configuration dataChangePublisher (< 100 users)	37
yab configuration dataChangePublisher (100-500 users)	37
yab configuration dataChangePublisher (> 500 users)	37
yab configuration jms queue	38
Query Service Tuning Parameters	39
yab configuration scheduledRefresh	39
yab configuration batchSize	39
JDBC Connections and Hikari	40
yab configuration hikari pool	40
OS Tuning	40
Transparent Huge Pages (THP)	40
Dirty Page Writeback (vm.dirty_bytes)	42
vm.swappiness	43
Network Offload (LRO/GRO)	45
ulimits (nproc, nofile)	46
Appendix A – Reference Material	47
Appendix B – PASOE	48
Introduction	48
PASOE architecture	48
YAB settings and for PASOE	50
ABL Sessions & Multi-Session Agents (Min / Max / Initial / Clients)	50
PASOE ABL Session lifecycle management (“Trimming”)	51
Appendix C - Requirements for an optimally tuned, scalable and stable system	52
OpenEdge Index Maintenance	52
Impacts	52
Java metaspace	53
Impact and Reference	53
Appendix D - Revision History	54

Introduction

This document is intended to serve as a technical tuning guide for QAD Adaptive 2608 Production Servers. It is assumed that appropriate sized hardware has already been selected and commissioned. For further information on sizing QAD Adaptive 2608 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 -omsiz 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.

Change Control - Important General Principles

Validation Process before making any Memory Related Changes

Before modifying any memory-related tuning parameter (heap sizes, buffer pools, cache allocations), the total committed memory demand on the server must be reviewed against available physical RAM. Increasing a single component's allocation without checking total system demand can directly cause a capacity outage.

The Total System Memory Budget is the sum of all committed memory consumers on the server. If this sum exceeds 85% of physical RAM, the change must not proceed without either adding RAM or reducing another component.

Area	Detail
Impact / Why Tune	Prevents memory exhaustion outages caused by individual component changes made without considering total system demand. Any heap or cache increase must be validated against this budget.
Formula	Total Committed = sum(all Java -Xmx) + MongoDB cacheSizeGB + sum(DB -B × blocksize) + OS/kernel overhead (4–6 GB). Must be < 85% of Physical RAM.
When to Apply	Before ANY change to: Tomcat heap (-Xmx), MongoDB cacheSizeGB, database buffer pool (-B), PASOE agent count, or any other memory-consuming parameter.
Responsibility	This is a process requirement — part of any RFC or change request that modifies memory allocation on a production server.

Table: Total System Memory Budget — review process

Safety Considerations for Runtime Changes: Red-Amber-Green Classification

When applying kernel or application tuning parameters to a running production system, each parameter must be assessed for its runtime impact. Changes that are safe on a fresh boot may be disruptive on a running system which has its own existing memory and runtime state. The following table classifies such changes into a Red-Amber-Green system.

Area	Detail
Green (safe at runtime)	Affects future operations only. No retroactive impact on current system state. Apply immediately and persist. Examples: Transparent Huge Pages, vm.swappiness, network offload.
Amber (requires restart)	Requires service or process restart to take effect, or has bounded runtime impact. Apply to config, restart at next maintenance window. Examples: ulimits, MongoDB cacheSizeGB, PASOE jemalloc, NiFi settings.
Red (sensitive — risk of disruption)	New value may conflict with current system state. Can cause I/O storms, out-of-memory conditions, or global stalls if current state exceeds new threshold. Apply to config only, let take effect at next planned reboot. Examples: vm.dirty_bytes, vm.min_free_kbytes.
General Rule	If uncertain about the current system state relative to a new parameter value, persist the change and let it take effect at the next planned reboot.

Table: Runtime change safety classification

Making Tuning Changes

All tuning changes described in this document are described in a way that covers what a standard change looks like. If the change needs to be made at runtime or in a Golden Image, then the changes should be made in an appropriate way. This document does not document all the different ways changes can be implemented, that is the responsibility of the party making the changes.

OS RAM Tuning dirty_bytes

The majority of servers are using the dirty_ratio of 20 or 30% of RAM for dirty pages. The default will be enabled if the following is 0:

```
sudo sysctl vm.dirty_bytes
```

```
vm.dirty_bytes = 0
```

Please change the following parameters online and also make them permanent.

```
vm.dirty_bytes=1073741824
```

```
vm.dirty_background_bytes=268435456
```

```
vm.min_free_kbytes=1048576
```

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.

-lruskip 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 -lruskip 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

-lruskip

- 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 lruskip

#Database lruskip

dbserver._base.lruskip=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 the 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 four (4) after-image fixed extents.
 - The default size should be 256MB for mfgdb (+any other busy database) and 128MB for all other databases, but always use QAD Monitoring to help size more accurately.
 - Ensure the AI Archiver policy is to archive each extent after ten (10) minutes (i.e. 600 seconds).
 - Set -aibufs to at least 1.5 times the number of -bibufs.

- Use QAD Monitoring to determine whether -aibufs should be increased further.
- 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

Table	Records	Size	-Record Size (B)-			---Fragments---		Scatter
			Min	Max	Mean	Count	Factor	
PUB.ih_hist	80279	32.4M	352	518	422	95599	1.2	6.6
PUB.ipb_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.

Please set `-hash (dbserver.INSTANCE.hashtableentries)` as a next prime number to the `-B` value.

Please set the `hashLatchFactor` to 100:

`dbserver._base.property.hlf.key=hashLatchFactor`

`dbserver._base.property.hlf.value=100`

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] -lrskips 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
<p>Create a batch job, run once per week, to apply online IDXCOMPACT at 85% to all indexes that breach the defined threshold</p> <ul style="list-style-type: none"> ● Set a threshold for indexes to be compacted ● Specifically: <ul style="list-style-type: none"> ○ 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 <p>Notes:</p> <ol style="list-style-type: none"> 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
```

PASOE OpenEdge 12 Tuning

PASOE OpenEdge 12 - PROD Tuning Guidelines

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
pas.INSTANCE.jvm.maxheap	maxHeap
pas._base.jvm.heapdumpoonoom	heapDumpOnOOM
pas._base.jvm.heapdumpppath	heapDumpPath
pas._base.jvm.gclog	GCLog

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.

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	1/1/1/1	1/1/1/1	1/1/1/1	2/2/2/1
minAgent	1/1/1/1	1/1/1/1	1/1/1/1	2/2/2/1
maxAgent	3/3/3/3	3/3/3/3	3/3/3/3	6/6/6/3
initABLSess	3/3/3/1	5/5/5/1	10/10/10/1	10/10/10/1
minAvailABLSess	3/3/3/1	5/5/5/1	10/10/10/1	10/10/10/1
maxABLAgentSess	10/10/10/5	15/15/15/8	20/20/20/8	30/30/30/8
maxAgentConn	10/10/10/5	15/15/15/8	20/20/20/8	30/30/30/8
maxHeap	512m/512m/512m /128m	512m/512m/512m/1 28m	1024m/1024m/102 4m/256m	2048m/2048m/20 48m/512m
	Max concurrent ABL Sessions (i.e. client requests)			
	30/30/30/15	45/45/45/24	60/60/60/24	180/180/180/24

NOTE 1: If the customer is heavily using other PAS instances such as eam, qxtnative or other pas instances please tune as per the qra / fin / mfg instances

Please ensure the following PAS configuration is also enabled:

pas._base.jvm.heapdumpoom=/^XX:+HeapDumpOnOutOfMemoryError.\$/-XX:+HeapDumpOnOutOfMemoryError

pas._base.jvm.heapdumppath=/^XX:HeapDumpPath=.\$/-XX:HeapDumpPath=\${base}/logs

```
pas._base.jvm.gclog=/^Xlog:gc.$/-Xlog:gc=info:file=${base}/logs/${name}-gc.log:time,uptime,level,tags:filecount=10,filesize=100M
```

Full details of exact YAB PAS Settings for all configurations can be found in the following googlesheet: [QAD Adaptive Tuning Guide OE12 PAS Tuning Details - 2025/2026](#) (unchanged)

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 -h 25 -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 -h 25
```

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 -h 25
```

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 -h 25
```

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 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.

PASOE OpenEdge 12 jemalloc

- In order to reduce the RAM demands and reclaim free memory for the _mproapsv agent it is recommended to implement jemallow for the PASOE instances as follows:

```
$ sudo su
# sudo dnf install jemalloc -y
# ls -l /usr/lib64/libjemalloc.so.2
### ensure that $DLC/bin/_mproapsv has 755 permission (it should not have the setuid bit)
# ls -ltrh /qond/apps/dlc/bin/_mproapsv
-rwxr-xr-x 1 root root 20M Aug 11 21:01 /qond/apps/dlc/bin/_mproapsv
(if required, update the file permission for $DLC/bin/_mproapsv to 755 in order to remove the setuid bit)
# exit
$ sudo mkdir /backup/<ticket#>
$ sudo chmod 777 /backup/<ticket#>
$ sudo -u mfg /qond/apps/dlc/bin/proenv
proenv> cp -p /qond/apps/mfgpro/build/config/configuration.properties /backup/<ticket#>
proenv> vi /qond/apps/mfgpro/build/config/configuration.properties
Append:
## <od***> - <date> - <ticket#>
pas._base.env.modules_run_quarantine.key=MODULES_RUN_QUARANTINE
pas._base.env.modules_run_quarantine.value=LD_LIBRARY_PATH
pas._base.env.ld_preload.key=LD_PRELOAD
pas._base.env.ld_preload.value=/usr/lib64/libjemalloc.so.2
pas._base-classic-ws.env.modules_run_quarantine.key=MODULES_RUN_QUARANTINE
pas._base-classic-ws.env.modules_run_quarantine.value=LD_LIBRARY_PATH
pas._base-classic-ws.env.ld_preload.key=LD_PRELOAD
pas._base-classic-ws.env.ld_preload.value=/usr/lib64/libjemalloc.so.2
<Esc>:wq
proenv> yab -a:/qond/apps/mfgpro pas-script
proenv> egrep "LD_PRELOAD|MODULES_RUN_QUARANTINE" /qond/apps/mfgpro/scripts/pas-* -start
proenv> exit
### Wait for the pas-* to restart
$ sudo cat /proc/<pas-*_agent_PID>/environ | tr '\0' '\n' | grep LD_PRELOAD
### You should see the following output:
```

```
LD_PRELOAD=/usr/lib64/libjemalloc.so.2
```

```
$ sudo /usr/bin/lsof -p <pas-*_agent_PID> | grep jemalloc
```

```
### You should see the following output:
```

```
_mproapsv <pas-*_agent_PID>  mfg  mem      REG          253,0      574800  16797843
/usr/lib64/libjemalloc.so.2
```

```
$ exit
```

PASOE OpenEdge 12 - 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. 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	2/2/2
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
maxHeap	512m, 512m, 512m, 128m	512m, 512m, 512m, 128m
	Max concurrent ABL Sessions (i.e. client requests)	
	10/5/5	30/16/16

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.

PASOE OpenEdge 12 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
```

The yab enhancement to use `terminateFreeABLSession` is coming in [RFRD-12373](#)

Further details on PASOE ABL Session lifecycle management is outlined in [Appendix B](#).

Specific QAD Application Tuning

MongoDB cacheSizeGB

MongoDB uses the WiredTiger storage engine, which maintains an internal cache for frequently accessed data. By default, WiredTiger sets this cache to $(\text{RAM} - 1 \text{ GB}) / 2$ — a formula designed for dedicated MongoDB servers. On shared QAD Adaptive systems where MongoDB runs alongside the full ERP stack, Logi Composer services, and multiple Tomcat instances, this default allows MongoDB to consume up to 50% of available RAM without any configured limit.

All QAD servers running MongoDB must have an explicit cacheSizeGB configured. The MongoDB cacheSizeGB setting must never be left at the default on shared systems. Logi Composer performance testing showed that MongoDB usage rarely exceeds 512 MB under normal QAD Action Center workloads — the recommended values below provide adequate headroom while preventing the runaway default behaviour.

Area	Detail
Impact / Why Tune	Without explicit sizing, MongoDB defaults to $(\text{RAM}-1)/2$ which dominates shared QAD systems. Must be explicitly configured. (REQUIREMENT)
Runtime Safety	Amber — requires MongoDB restart. The cacheSizeGB setting is read at mongod startup only. Apply at next maintenance window.
Permanent Config	yab configuration: <code>mongodb._base.properties.storage.wiredTiger.engineConfig.cacheSizeGB=X.</code> Set in configuration.properties, then: <code>yab reconfigure;</code> <code>yab mongodb-qad-collaboration-stop;</code> <code>yab mongodb-qad-collaboration-start</code>
Online Execution	Not applicable — requires MongoDB restart.
Load / Size Dependency	Size-dependent — see following table for values per customer tier.

Table: MongoDB cacheSizeGB — implementation summary (tuning is load/size dependent, see following table for details)

Tier	Typical Server RAM	cacheSizeGB	Formula
Small (1–49 users)	32 GB	2	~5% of RAM
Medium (50–199 users)	64 GB	3	~5% of RAM
Large (200–499 users)	128 GB	6	~5% of RAM
A+ (500+ users)	256 GB	12	~5%, capped

Table: MongoDB cacheSizeGB — values by customer tier

PASOE jemalloc

The Progress Application Server for OpenEdge (PASOE) uses multi-session agents (_mproapsv processes) to execute ABL business logic. Under the default glibc memory allocator, these agents exhibit unbounded memory growth over time due to heap fragmentation — freed memory is not efficiently returned to the operating system.

The jemalloc allocator resolves this by managing memory in thread-local arenas with aggressive free-page purging. All OE12 PASOE instances must run with jemalloc loaded via LD_PRELOAD. This has been deployed fleet-wide on Cloud since 2023 and is explicitly required for the 2608 release.

Area	Detail
Impact / Why Tune	Without jemalloc, PASOE agents grow unbounded in memory over time (glibc fragmentation). Requires frequent restarts to reclaim RAM. (REQUIREMENT for OE12)
Runtime Safety	Amber — requires PASOE restart. The LD_PRELOAD environment variable is read at process start only.
Permanent Config	<p>This is an example way of adding this as a permanent configuration</p> <p>yab configuration (use underscores, not hyphens):</p> <pre>pas._base.env.modules_run_quarantine.key=MODULES_RUN_QUARANTINE; pas._base.env.modules_run_quarantine.value=LD_LIBRARY_PATH; pas._base.env.ld_preload.key=LD_PRELOAD; pas._base.env.ld_preload.value=/usr/lib64/libjemalloc.so.2.</pre> <p>Set in configuration.properties, then:</p> <p>restart PASOE instances.</p> <p>CONFIRM that the parameters are active after the restart</p>
Prerequisites	<p>These are the prerequisite installs, it can be done in an appropriate way on your system:</p> <p>Install jemalloc:</p> <pre>sudo dnf install jemalloc -y.</pre> <p>Verify:</p> <pre>ls -l /usr/lib64/libjemalloc.so.2.</pre> <p>Ensure \$DLC/bin/_mproapsv has 755 permissions (not 4755 or 6755 — the setuid bit must not be set, as the kernel ignores LD_PRELOAD for setuid binaries).</p>
Online Execution	Not applicable — requires PASOE restart.
Load / Size Dependency	None — applies to all OE12 servers with PASOE regardless of user count.

Table: PASOE jemalloc — implementation summary

NiFi Content Repository Archive Percentage

Apache NiFi uses a content repository to store flow file content during processing. By default, NiFi stops archiving (and effectively stops processing) when the disk device hosting the content repository reaches approximately 52% utilisation. This causes a silent failure — background processing stops without any alert or visible error to users.

Setting the archive maximum usage percentage to 90% prevents this silent failure by allowing NiFi to continue operating until disk usage reaches a much higher threshold, at which point standard disk space alerts would already have fired.

Area	Detail
Impact / Why Tune	Without this setting, background processing silently stops at 52% disk utilisation. No alert fires, no error is visible. Data flow halts without notification. (Recommend elevation to REQUIREMENT)
Runtime Safety	Amber — requires NiFi restart to pick up configuration change.
Permanent Config	yab configuration: nifi.default.application.nifi.content.repository.archive.max.usage.percentage=90%
Online Execution	Not applicable — requires NiFi restart.
Load / Size Dependency	None — applies to all servers running NiFi.

Table: NiFi content repository archive percentage — implementation summary

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.startuptopts=-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.startuptopts=-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.startuptopts=-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.startuptopts=-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 TuningXmx

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.

OS Tuning

Transparent Huge Pages (THP)

Transparent Huge Pages (THP) is a Linux kernel memory management feature that automatically promotes standard 4 KB pages to 2 MB huge pages. While intended to reduce Translation Lookaside Buffer (TLB) overhead, Transparent Huge Pages causes memory allocation latency, fragmentation, and unpredictable memory bloat on database and JVM workloads.

Progress do not support Transparent Huge Pages on systems running OpenEdge databases. MongoDB recommends disabling THP for all versions prior to 8.0. Both of these components are present on standard QAD Adaptive servers. Disabling Transparent Huge Pages is a vendor-backed requirement.

Area	Detail
Impact / Why Tune	Causes unpredictable memory bloat and allocation latency on database workloads. Must be disabled on all QAD servers. (REQUIREMENT)
Runtime Safety	Green — safe to apply at runtime. Kernel stops using THP for new allocations immediately. Existing huge pages remain until freed naturally. No I/O spike or memory rebalancing triggered.
Permanent Config	<p>Persist via the appropriate OS configuration mechanism (Salt state, golden image, or sysctl.d — coordinate with Cloud Ops for the standard deployment path)</p> <p>GRUB: Add transparent_hugepage=never to GRUB_CMDLINE_LINUX in /etc/default/grub. Then: grub2-mkconfig -o /boot/grub2/grub.cfg.</p> <p>Or: Salt state rocky.adaptive_tuning</p> <p>defrag has no kernel boot parameter equivalent — it must be set separately</p> <p>Options for persisting defrag=never:</p> <ol style="list-style-type: none"> 1. tuned profile — a custom tuned profile can set both: <pre> [sysfs] /sys/kernel/mm/transparent_hugepage/enabled=never /sys/kernel/mm/transparent_hugepage/defrag=never </pre> 2. systemd unit (tmpfiles.d or a oneshot service at boot): <pre> # /etc/systemd/system/disable-thp.service [Service] Type=oneshot ExecStart=/bin/sh -c 'echo never > /sys/kernel/mm/transparent_hugepage/enabled && echo never > /sys/kernel/mm/transparent_hugepage/defrag' </pre>
Online Execution	<pre> echo never > /sys/kernel/mm/transparent_hugepage/enabled; echo never > /sys/kernel/mm/transparent_hugepage/defrag </pre>
Load / Size	None — applies identically to all server sizes and all user counts.

Dependency

Table: THP tuning — implementation summary**Dirty Page Writeback (vm.dirty_bytes)**

The dirty page writeback subsystem controls how much modified data the kernel allows to accumulate in memory before flushing it to disk. By default, Linux uses a percentage-based threshold (vm.dirty_ratio, typically 20–30% of RAM). On large-memory servers this allows tens of gigabytes of dirty data to accumulate, creating sudden write bursts that saturate storage and stall applications.

Using absolute byte thresholds (vm.dirty_bytes and vm.dirty_background_bytes) instead of percentages ensures predictable, consistent writeback behaviour regardless of server RAM size. The dirty page writeback settings interact with the memory reclaim subsystem — vm.swappiness and vm.min_free_kbytes affect how the kernel manages dirty pages under memory pressure.

Area	Detail
Impact / Why Tune	Prevents I/O stalls caused by large dirty page accumulation. Without absolute thresholds, a 128 GB server allows up to 25 GB of dirty data (20% default) before forcing synchronous writes — causing multi-second application stalls.
Runtime Safety	Red — if current dirty pages exceed the new threshold, the kernel immediately throttles ALL writes until dirty pages drop below the limit. Check current state before applying at runtime: <code>grep Dirty /proc/meminfo</code> . If Dirty > 1 GB, defer to next reboot.
Permanent Config	Persist via the appropriate OS configuration mechanism (Salt state, golden image, or <code>sysctl.d</code> — coordinate with Cloud Ops for the standard deployment path) Takes effect at next reboot.
Online Execution	<code>sysctl -w vm.dirty_bytes=1073741824;</code> <code>sysctl -w vm.dirty_background_bytes=268435456.</code> ONLY if current Dirty value in <code>/proc/meminfo</code> is below 1 GB.
Load / Size Dependenc	None — applies identically to all server sizes. The absolute values (1 GB foreground, 256 MB background) are appropriate for all QAD server configurations.

y

Table: Dirty page writeback tuning — implementation summary

vm.swappiness

The `vm.swappiness` parameter controls the kernel's preference for reclaiming memory from anonymous pages (process heap/stack — requires swap I/O) versus file-backed pages (page cache — can be discarded and re-read). A lower value reduces swap activity, keeping database working sets and JVM heaps resident in physical RAM.

Setting `vm.swappiness` to 1 is the recommended value. A value of 0 completely disables anonymous page reclaim, which can trigger the out-of-memory killer directly without gradual degradation. A value of 1 allows minimal swap activity as a last resort before the out-of-memory killer is invoked.

Note: `vm.swappiness=1` creates cliff behaviour when combined with memory overcommit. The system operates normally until available RAM is exhausted, then collapses instantly with no gradual degradation. This is why the Total System Memory Budget (above) is a prerequisite — without adequate headroom, low swappiness masks the overcommit problem until it becomes catastrophic.

Area	Detail
Impact / Why Tune	Keeps database working sets (OpenEdge buffer pools, JVM heaps) resident in RAM. Prevents latency spikes from swap I/O that degrade application responsiveness.
Runtime Safety	Green — safe to change at runtime. Affects future page reclaim decisions only. Current swap content is unchanged. Full benefit realized after a planned restart clears existing swap.
Permanent Config	Persist via the appropriate OS configuration mechanism (Salt state, golden image, or <code>sysctl.d</code> — coordinate with Cloud Ops for the standard deployment path)
Online Execution	<code>sysctl -w vm.swappiness=1</code>
Load / Size Dependency	None — applies identically to all server sizes and all user counts.

Table: vm.swappiness tuning — implementation summary

Network Offload (LRO/GRO)

Large Receive Offload (LRO) and Generic Receive Offload (GRO) are network interface features that coalesce multiple incoming TCP packets into larger buffers before passing them to the kernel. While this reduces CPU overhead for bulk data transfer, it breaks the precise TCP segment boundaries that Progress OpenEdge database communication relies upon.

Progress require Large Receive Offload and Generic Receive Offload features disabled for correct OpenEdge operation. This has been a Cloud Operations standard since 2015 and applies to all QAD servers, particularly on virtualised platforms where offloading behaviour is inconsistent.

Area	Detail
Impact / Why Tune	Progress OpenEdge requires precise TCP segment boundaries. Network offloading coalesces packets, breaking Progress protocol expectations and causing connection errors.
Runtime Safety	Green — safe to disable at runtime. Active connections are not disrupted. Change takes immediate effect for new packet processing.
Permanent Config	Persist via the appropriate OS configuration mechanism (Salt state, golden image, or sysctl.d — coordinate with Cloud Ops for the standard deployment path) Add to /etc/rc.local: ethtool -K eth0 lro off gro off
Online Execution	NIC=\$(ip route show default awk '/default/{print \$5; exit}'); ethtool -K \${NIC} lro off gro off
Load / Size Dependency	None — applies identically to all server sizes.

Table: Network offload tuning — implementation summary

ulimits (nproc, nofile)

The ulimit settings control per-user resource limits for maximum number of processes (nproc) and maximum open file descriptors (nofile). Insufficient limits cause appserver startup failures, "too many open files" errors, and connection refused conditions under load.

These ulimit values have been standard on all Cloud servers since 2015 and are checked by the `validate_config.sh` automation tool. They are included here because non-Cloud, Lift+Shift, and manually provisioned environments may lack these settings.

Area	Detail
Impact / Why Tune	Insufficient limits cause appserver startup failures and "too many open files" errors under production load.
Runtime Safety	Amber — config change is immediate but running processes retain their existing limits until restarted. Application stack restart required for processes to pick up new values.
Permanent Config	Add to <code>/etc/security/limits.conf</code> : * soft nproc 65536; * hard nproc 65536; * soft nofile 1048576; * hard nofile 1048576
Online Execution	Not applicable — running processes require restart to inherit new limits.
Load / Size Dependency	None — applies identically to all server sizes and all user counts.

Table: ulimits tuning — implementation summary

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](#)

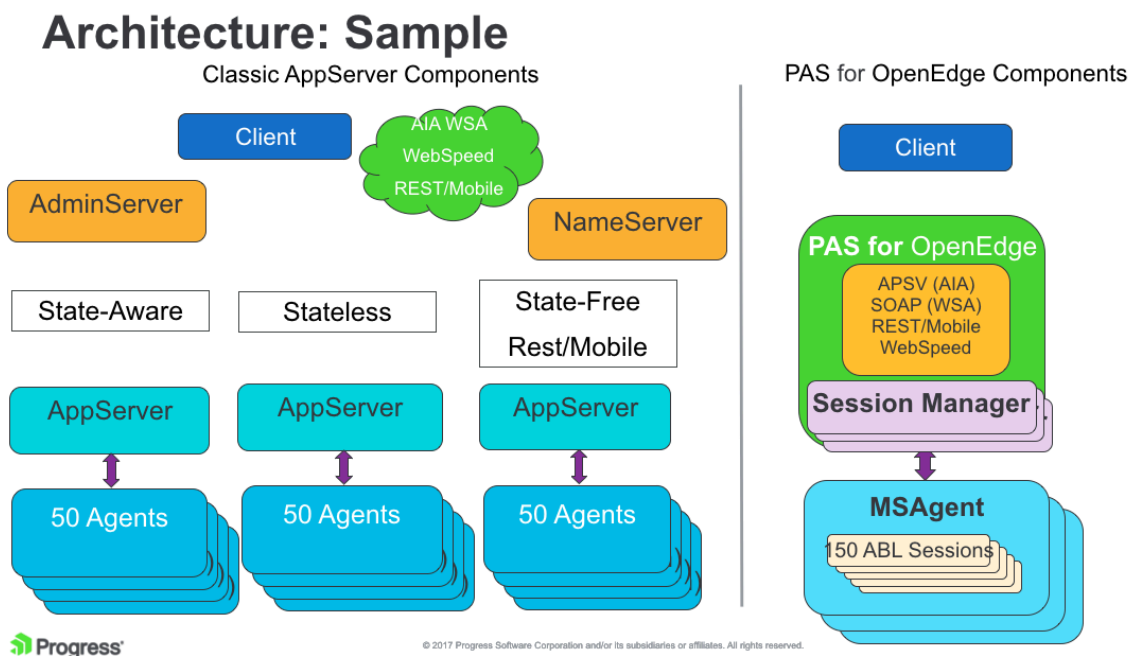
Appendix B – PASOE

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.



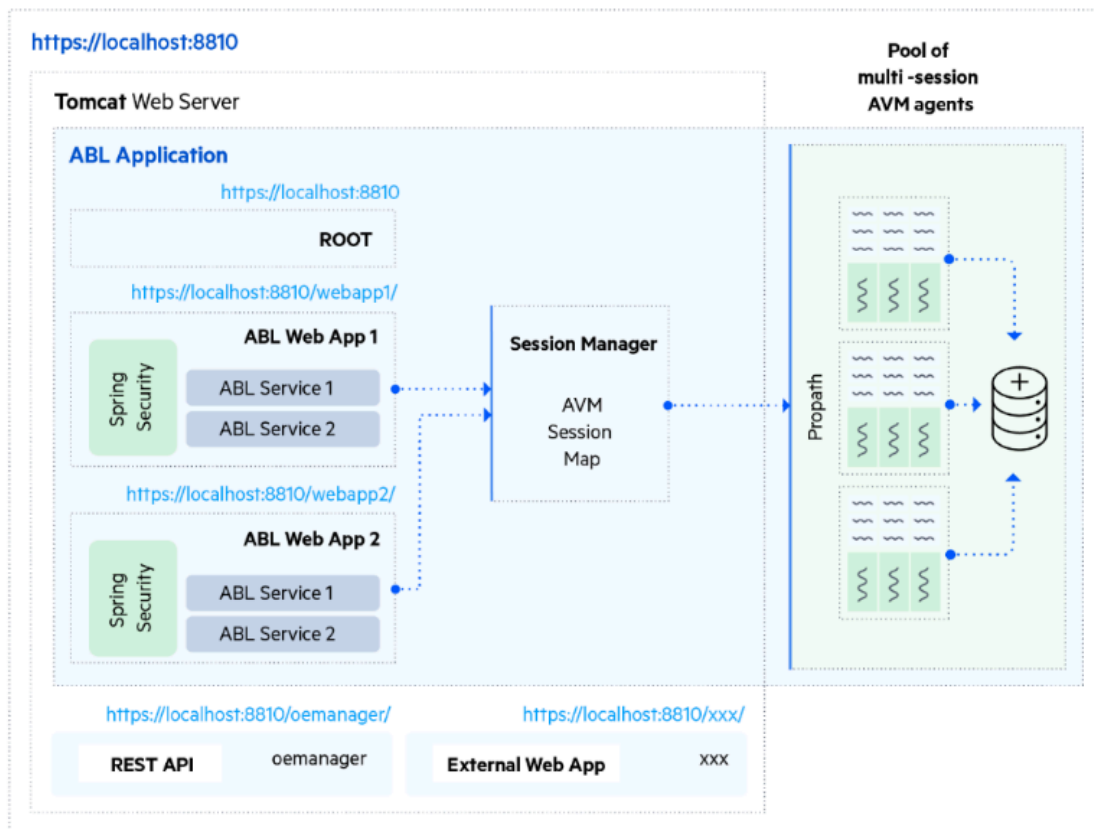


Figure: The main components of each PASOE instance (see the [Progress documentation](#)).

- 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.

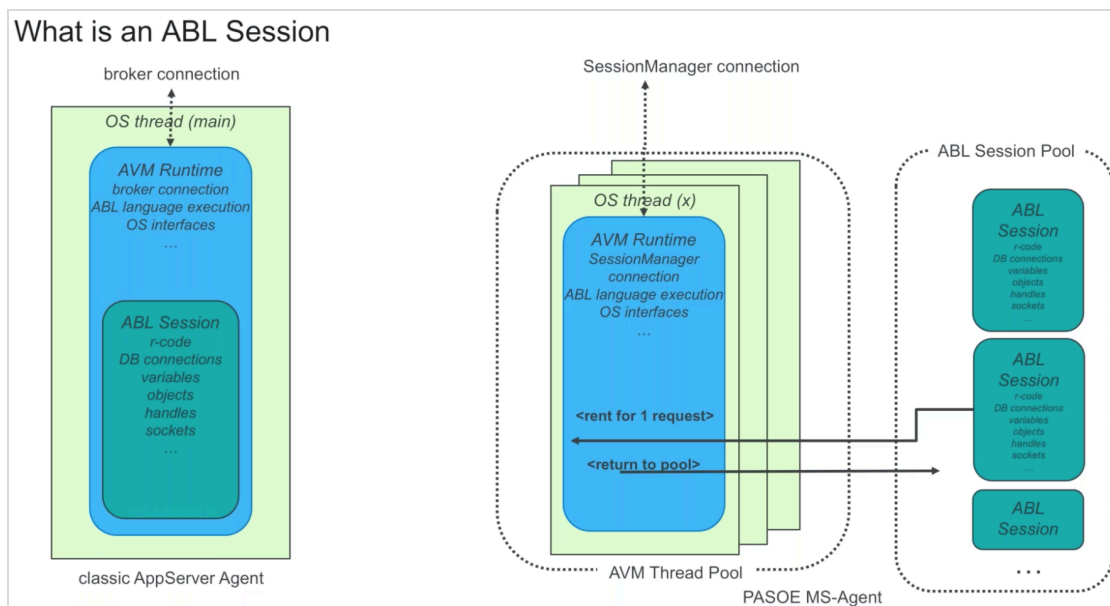


Figure: Each PASOE appserver can have one or more Multi-Session agents configured, where each MSA contains a (configurable) pool of ABL Sessions.

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	<ol style="list-style-type: none"> 1. Always retain at “minAgent” MSA per PASOE appserver <ol style="list-style-type: none"> a. minAgent by default should be 1, i.e. never trim ALL MSAs b. E.g. minAgent=1, # active MSAs= 2, then we have 1 MSA potentially available to trim 2. Trimm the MSAs if: <ol style="list-style-type: none"> 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. trim 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 - maxSrvInstances
- 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-2021 parameters

Version 6.2

- Migrated from Sept-2020 to March-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

Version 12.1

- PAS Tuning changed based on customer production envs
- Added OS tuning for dirty_pages
- Added jemallow for PAS Instances

Version 13.0

- Created version for Adaptive 2608
- Tidied up Appendix
- Added new tuning section and sections to align with current standards