

PASOE

A Game of Memory

How does it work?

Korak Schoone

Senior Consultant, Progress Professional Services



**How does the PASOE deal
with memory?**

And how do I manage it?

Where is the memory used?

- Two or more system processes are running for a PASOE instance
 - A Java process for the Tomcat Application server
 - Hosts the web apps and ABL Session manager app
 - One or more Multisession agents (`_mproapsv`) per ABL application on the PASOE instance
 - Hosts the ABL sessions to run ABL code
 - Will be the big memory user of the PASOE



Why would I want to know the memory usage?

- Right-size the machine resources
- Monitor it to prevent it from being killed by OOM reaper (Linux)
- Prevent Memory pressure on the host machine, which could adversely affect the performance



MS Agent (_mproapsv)

- Multi-threaded server process
- Handles multiple request simultaneously
- Each request executes with the context of an ABL session
- ABL sessions in the agent run in isolation from each other
- Think of a single ABL session as a ABL client batch process (_progress -b)
- Each ABL session keeps track of the memory allocated to it
 - Can be requested via OEM/oemanager API/OEJMX

MS Agent Overhead

- Agent uses more memory than the ABL session memory combined
- Agent also runs to some special ABL sessions to manage all running sessions (can be seen in agent log file)
 - AS-Listener – Single dedicated thread that listens for request coming from Tomcat
 - AS-ResourceMgr – Single dedicated thread that manages resources shared by all Session (f.i. DB connections when using Shared Memory connections)
 - AS-Admin – Single dedicated thread for handling administrative request, f.i. getting metrics, stopping a session. It also collects metrics, so needs memory to store those

MS Agent Reporting

- PASOE tracks allocation and deallocation of memory for all ABL sessions and its overhead
- Can report this information through OEJMX or REST api
- Cannot report on memory allocated to underlying processes, such as
 - 3rd party libraries (OpenSSL, XML parser, etc)
 - System calls (f.i, print buffers)
- Still a good representation as most memory will be by the Agent itself

Factors in the OS accounting of Memory Use

Swap

- OS manages memory in 4k/8k/ chunks (pages)
- If total memory usage on the machine approaches physical limit pages will be written to swap
- Swap is a file on disk and disk access is slow
- When page is needed, it is read back in memory (by replacing another page)
- This process will negatively impact performance of all process on the machine

Swap - PASOE

- If the PASOE has a lot of memory in Swap, there are generally 2 reasons:
 - The machine is using more memory than physical available (all processes combined)
 - PASOE has memory allocated that has not been used in a while (and thus those pages are candidates for swapping)
- Consider to:
 - Increase the available physical memory
 - Reduce the memory needed by the application
 - Reduce footprint of PASOE

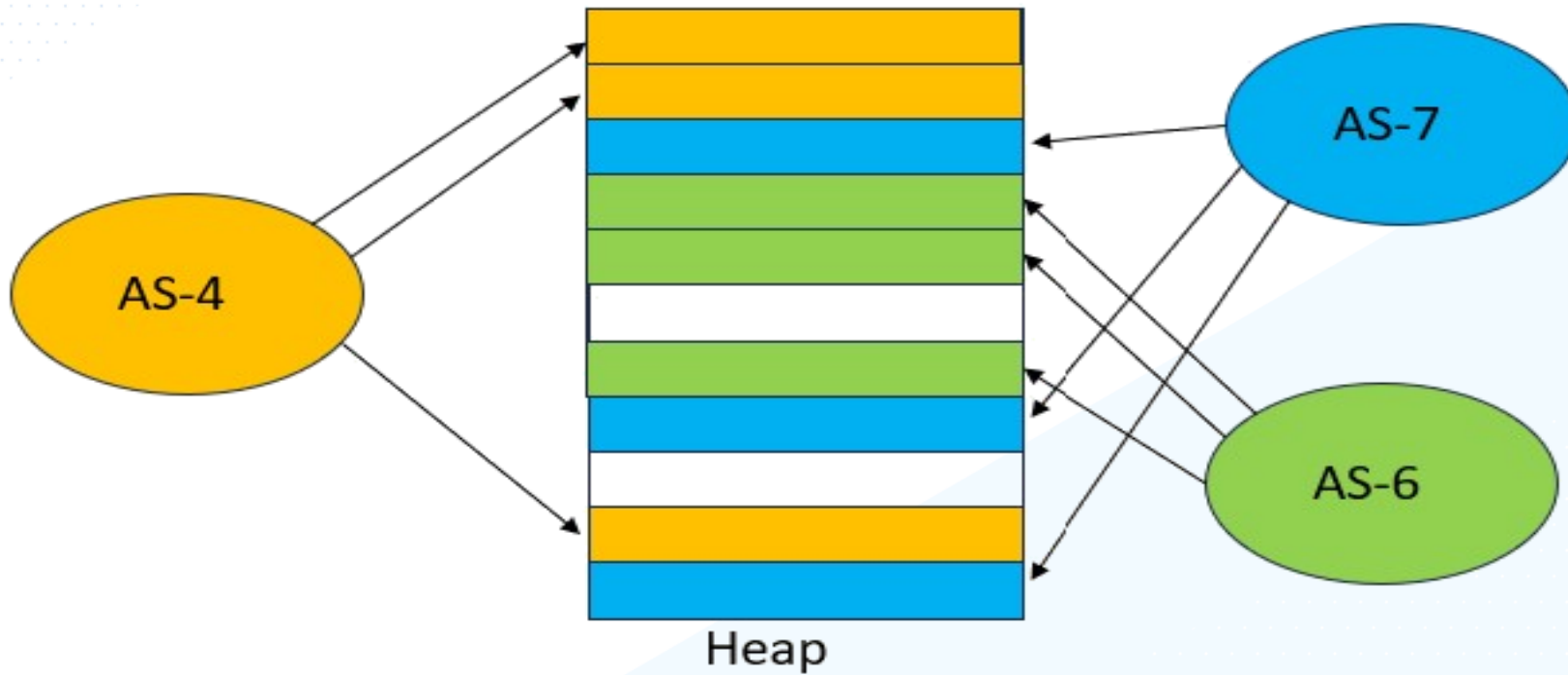
The Heap

- The Heap represents the dynamically allocated memory to a process
- Each process has its own Heap
- Is the memory used for its data needs
- Does not include the executable image, shared libraries, static data segment or the stack
- Managed by the malloc API
- Can grow based on demand. Asks OS to extend the heap. This is a relatively expensive operation
- If there are 10 ABL session on the agent of 50Mb each, the Heap will be 0.5 Gb

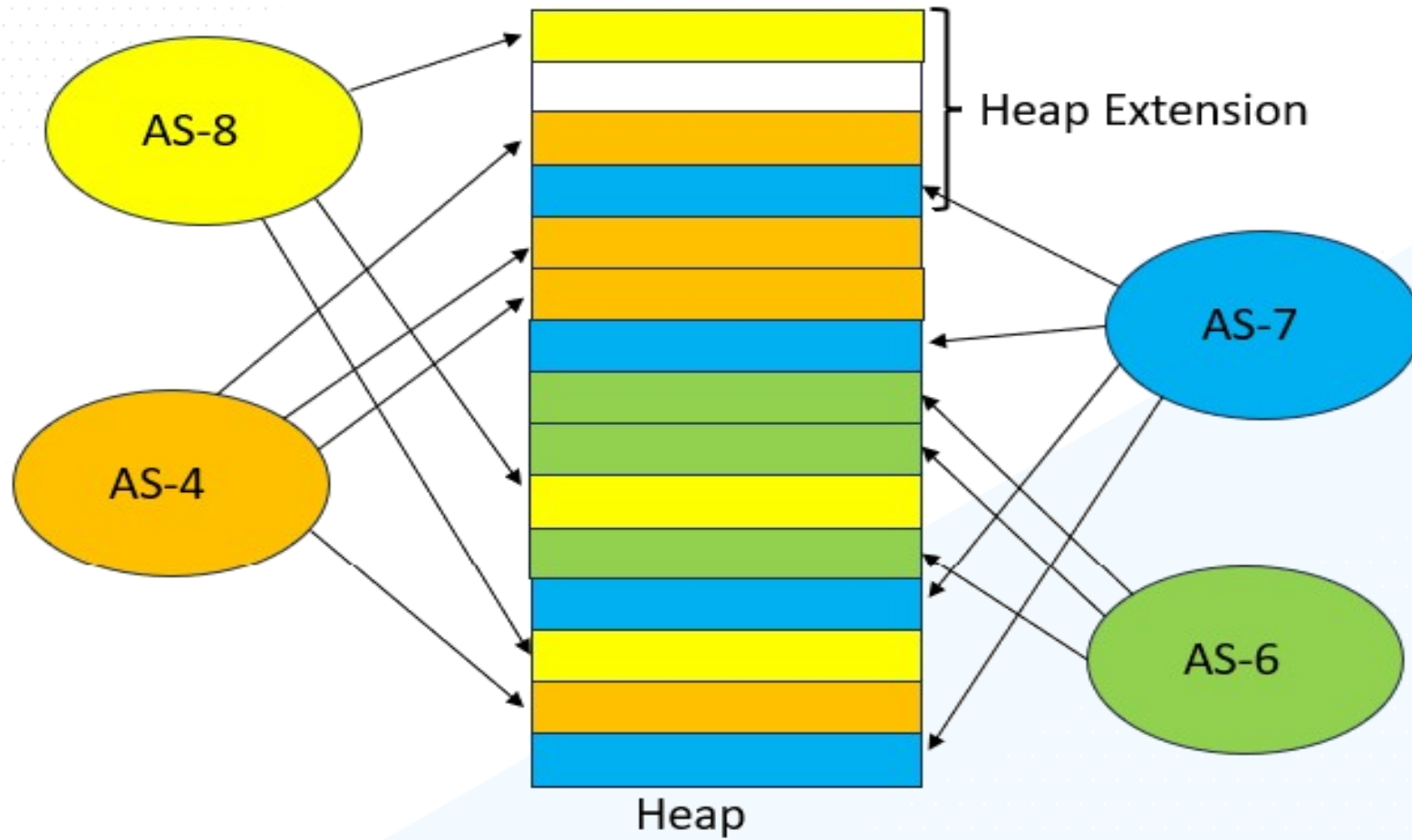
The Heap - Fragmentation

- Memory is allocated to the OS process (Agent)
- Agent assigns (parts of) the Heap to an ABL session
- If a Session gets terminated, the used memory is released to the Agents Heap
- Memory from the Heap be used by multiple Sessions
- Stopping Sessions might not release memory to the OS, as that can only happen if the top of the heap (the memory extension) is fully not in use
- The malloc API focusses on efficiency of time and space. It does not reorganize (defrag) the memory as this is a heavy process

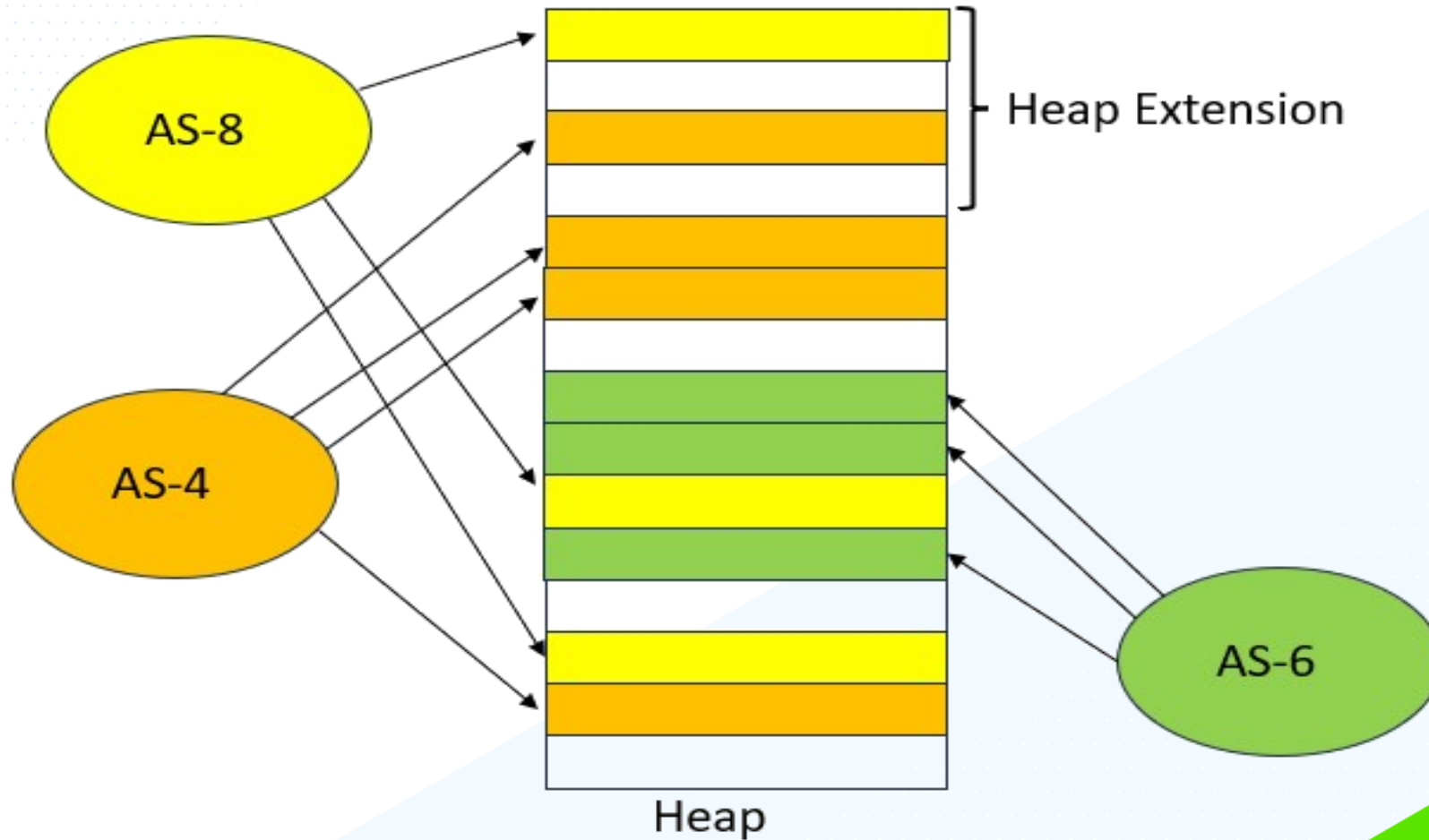
The Heap – Simplified visualization



The Heap – Simplified visualisation



The Heap – Simplified visualisation



The Heap will grow to meet the highest demand, but then never (or rarely) shrink back

OS-COMMAND & INPUT/OUTPUT THROUGH

- Linux / Unix only
- Triggers a fork() and exec()
- Creates a new child process, with the current memory footprint of the Agent (not just the session, as the agent is the OS process and owner of the memory)
- Makes a (full) copy of the memory of the agent, which blocks the memory during the copy
- Duplicates the memory usage on the machine (or more if multiple sessions are running this simultaneously)
- Memory is cleaned up if command is finished

OS-COMMAND & INPUT THROUGH

- Eliminate, consolidate and otherwise minimize the usage of OS-COMMAND and INPUT THROUGH in your application
- If not possible, configure the PASOE in a classic style (many agents with one or a couple of session max) to reduce the memory footprint of the agent
- <https://community.progress.com/s/article/pasoe-server-running-os-command-or-unix-statements-runs-very-slowly>

Memory Leaks

- Memory leaks can be in multiple components of the PASOE
 - Tomcat or anything hosted on it
 - There are not current know memory leaks in Tomcat or the OpenEdge components running on there
 - 3rd party or owned java applications hosted on the PASOE could contain memory leaks
 - Multi-session agent
 - There are no current know memory leaks in the agent code itself
 - ABL code from your application run on the MS Agent can contain memory leaks, which count towards the memory used by the Agent
 - External processes called from ABL code. These (usually) do not count towards the memory used by the agent

Dealing with Memory Leaks

- Best solution is to identify and address these memory leaks in the ABL application
- These can be caused by (not a complete list):
 - Persistent procedures initiated, but not deleted
 - Memptr variables not reset
 - ABL objects not cleaned by the garbage collector (circular reference)
 - Dynamically created objects (buffers, temp-tables, datasets) which are not deleted (these are never collected by the garbage collector)
 - Implicit created data objects (when receiving a dataset handle from the appserver f.i.)

Memory Leaks - Coding best practices

- Use FINALLY block to ensure cleanup
 - Always called after normal execution or return or thrown error
 - Run DELETE OBJECT from here on objects/handles
 - Be cautious with JSON objects inside larger structures
 - Cannot delete objects before they are returned as parameters
 - Remember to delete anything you create (explicitly and implicitly)
 - Deallocate anything that stores data for temporary use
 - MEMPTR, LONGCHAR

Memory Leaks - Coding best practices

- Create an unnamed WIDGET-POOL with explicit DELETE WIDGET-POOL
 - Yes, even classes offer a USE-WIDGET-POOL option
 - See: [KB#166227](#) "WIDGET-POOLS Explained"
- Pass temp-tables or datasets via handle then delete when finished
 - Make sure data is written out/returned successfully, first
 - Similarly, pass by-reference when possible

Memory Leaks – Mitigation

- Finding and removing the memory leaks from the code can be a long process, during which the application needs to stay running
- How can we do that?
- Pre-reap Agents (trimming agents for the classic Appserver)
 - Not a standard option
 - Use cron-job (or scheduler) to periodically check the memory usage of agents and use OEJMX or the oemanager API to gracefully shutdown agents that are using too much memory. New agents will be spawned if needed by the PASOE. Do not 'reap' all agents, as this might impact availability for a short moment

Memory Leaks – Mitigation

- From OE 12.7 there are new Session Life Cycle Management options available that can automatically trigger the trimming of a session.
- This will (probably) not return the memory to the OS, but will free up memory in the Heap to be reused by other (new) sessions
- Options to stop a request are:
 - After a number of request
 - If at the end of the request the memory exceeds a set limit
 - If during the request the memory exceeds a set limit there are two options
 - Finish the request
 - Raise a Stop condition during the request

Memory Leaks – Mitigation

conf\openedge.properties

[AppServer.Agent]

ablSessionRequestLimit=N

ablSessionMemoryLimit=N

ablSessionActiveMemoryLimitFinish=N

ablSessionActiveMemoryLimitStop=N

ablSessionMemoryDump=[0,1]

Identifying Memory Leaks

Tools

OEJMX / oemanager

OEJMX

- JMX is a Java technology that supplies tools for managing and monitoring applications, system objects, devices (such as printers) and service-oriented networks.
- Runs on Windows or Unix
- Runs locally only
- OEJMX is a set of classes to monitor and manage the OpenEdge functionality of the PASOE
- `<instance-name>bin\oejmx.bat(.sh)`

oemanager

- WebApp supplied with PASOE which hosts a set of REST services on top of the OEJMX functionality
- Not all OEJMX functions are available (yet) as a REST service
- Allows for remote use of the OEJMX services
- Swagger UI added to test the services (should be disabled in production)
- Can be called from any HTTP utility (curl, Postman, SoapUI, ABL HTTP client)
- Can be used to get information on the objects in memory of an ABL session

Demo: oemanager



LeakCheck

OpenEdge.Core.Util.leakCheck

- Logfile parser utility for client and appserver logs, based on the log-entry-type “DYNOBJECTS”
- Introduced in OpenEdge 12.6. Before some .p’s available in different knowledge base articles
- This utility is version dependent and maintained by Progress now and shipped with the release.

leakCheck – Log Entry Type

- Enable logging on the session
 - Client logging - -clientlog startup parameter
 - PASOE – Already enabled, entries in agent log
- Set log entry types to include "DynObjects.*[:<loglevel>]"
 - Client - -logentrytypes startup parameter
 - PASOE – openededge.properties – agentLogEntryTypes
 - If allowRuntimeUpdates is enabled, this can be adjusted while the PASOE is running
- Loglevel: 0 – 4 for increasing detail
 - Caution: level 4 can generate great amounts of log entries

leakCheck – Log File example

```
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1020 (ResolveToken OpenEdge.Core.Util.TokenResolver @ 626) Progress.Lang.Class
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1021 (ResolveToken OpenEdge.Core.Util.TokenResolver @ 626) OpenEdge.Logging.TokenResolverEventArgs
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created Progress.Lang.Object Handle:1022 (ResolveToken OpenEdge.Core.Util.TokenResolver @ 679) Progress.Lang.OERequestInfo
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1022 (ResolveToken OpenEdge.Core.Util.TokenResolver @ 679) Progress.Lang.OERequestInfo
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created Progress.Lang.Object Handle:1023 (ResolveToken OpenEdge.Core.Util.TokenResolver @ 692) Progress.Lang.OERequestInfo
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1023 (ResolveToken OpenEdge.Core.Util.TokenResolver @ 692) Progress.Lang.OERequestInfo
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1019 (Resolve OpenEdge.Core.Util.TokenResolver @ 812) OpenEdge.Core.Util.TokenResolverEventArgs
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1012 (Resolve OpenEdge.Logging.TokenResolver @ 87) OpenEdge.Core.Util.Token
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1013 (Resolve OpenEdge.Logging.TokenResolver @ 87) OpenEdge.Core.Util.Token
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created Progress.Lang.Object Handle:1024 (NamedFileWriter OpenEdge.Logging.Writer.NamedFileWriter @ 68) OpenEdge.Core.File
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created Progress.Lang.Object Handle:1025 (File OpenEdge.Core.File @ 129) OpenEdge.Core.Folder
2025-04-11T16:28:28.259+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created Progress.Lang.Object Handle:1026 (propGet_Logger OpenEdge.ApplicationServer.AgentInfo @ 104) OpenEdge.Logging.Logger
2025-04-11T16:28:28.260+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1002 (logMessage OpenEdge.ApplicationServer.AgentInfo @ 145) OpenEdge.Core.File
2025-04-11T16:28:28.260+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted-by-GC Progress.Lang.Object Handle:1005 (logMessage OpenEdge.ApplicationServer.AgentInfo @ 145) OpenEdge.Core.Folder
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created TEMP-TABLE Handle:1028 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created BUFFER Handle:1029 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT Table:_AgentStatHist
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created TEMP-TABLE Handle:1030 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created BUFFER Handle:1031 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT Table:_AgentThread
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created TEMP-TABLE Handle:1032 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created BUFFER Handle:1033 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT Table:_AgentSession
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created TEMP-TABLE Handle:1034 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created BUFFER Handle:1035 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT Table:_AgentConnection
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created TEMP-TABLE Handle:1036 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created BUFFER Handle:1037 (setArchiveAgentStats OpenEdge.ApplicationServer.AgentInfo @ 182) IMPLICIT Table:_AgentRequest
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created QUERY Handle:1038 (getAgentStatData OpenEdge.ApplicationServer.AgentInfo @ 203) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted QUERY Handle:1038 (getAgentStatData OpenEdge.ApplicationServer.AgentInfo @ 203) IMPLICIT
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Allocated LONGCHAR 0x7F3D0C1D56A0 size: 187 (getAgentStatData OpenEdge.ApplicationServer.AgentInfo @ 203)
2025-04-11T16:28:28.263+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deallocated LONGCHAR 0x7F3D0C1D56A0 size: 187 ( @ 0)
2025-04-11T16:28:28.268+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Created QUERY Handle:1039 (getAgentRequestQty OpenEdge.ApplicationServer.AgentInfo @ 540)
2025-04-11T16:28:28.268+0200 606571 606575 2 AS-Admin mtapsv:-? DYNOBJECTS Deleted QUERY Handle:1039 (getAgentRequestQty OpenEdge.ApplicationServer.AgentInfo @ 553)
```

leakCheck – Code example

```
using OpenEdge.Core.Util.LeakCheck.  
using Progress.Json.ObjectModel.JsonObject.  
  
var LeakCheck checker = new LeakCheck().  
var JsonObject report.  
  
checker:ParseLog('RestApi.agent.2025-04-11.log').  
  
if checker:HasLeaks() then  
do:  
    report = checker:GetReport().  
    report:WriteFile("LeakCheckReport.json").  
end.  
else message "There are no leaks found." view-as alert-box.
```

Demo: LeakCheck



Memory Profiler

OpenEdge Memory Profiler

- Detect and diagnose memory consumption issues of ABL applications (OpenEdge 12.8.3+)
- Identify Memory Leaks
- Enhance system and application stability
- Optimize performance
- Debug complex issues
- Optimize resource management
- Ensure best practices

OpenEdge Memory Profiler

- Reports memory consumption for a specific ABL session (ABL client, Batch session, PASOE ABL session)
- Each session has its own log file (.oemp)
- Log files need to be analyzed by Memory Profiler visualization tool
- Data is both Time-series (snapshots) and Relational
- Differentiates between:
 - Platform memory
 - Application memory

Enabling Memory Profiler - Session startup parameters

- `-profilememory <config-file>`
- All parameters for the memory profiler are in the config file
 - cadence N – How often (N = seconds) a regular snapshot is made (0 = off, default = 10)
 - Description *string* – To distinguish between reports
 - report-dir *directory* – storage folder of log files (default = working directory)
 - SnapshotEndOfRequest R – Frequency of `_EndOfRequest` snapshot. $R = 1 \rightarrow$ every request, $R = 10 \rightarrow$ every 10th request, 0 = off, default = 10
- Properties are case-insensitive
- `#` is a comment line

Enabling Memory Profiler - Config file example

regular snapshot every 5 seconds

cadence 5

description can be a single token or quoted string

description "PUG presentation"

report-dir can be fully qualified or relative to working directory of the PASOE

report-dir ..\..\profilelog

SnapshotEndOfRequest 10 is the default

Taking a Snapshot programmatically

- AVM must be enabled for memory profiling
- Custom tags can be assigned
- Allows to associate the data with places in the application code
- Tag can not begin with underscore (_). Reserved for automatic tags

USING Progress.Profile,MemoryProfiler.

MemoryProfiler:TakeSnapShot(“Activate procedure”)

Memory profile recording

- The recording is comprised of a series of snapshots
- Saved as a .oemp file
 - ABL client - memprof.*PID*.oemp (memprof.123456.oemp)
 - PASOE ABL session – memprof.*PID*.*AS-ID*.oemp (memprof.123456.AS-4.oemp)
- PID – OS process ID for the ABL session (prowin, pro) or the PASOE agent (_mproapsv)
- AS-ID – The session ID of a PASOE ABL session on the agent
- Format of the .oemp is not meant for human readability
- Session needs to be ended for .oemp to be analyzed

Analyzing Memory profile recording

- OpenEdge Memory profile visualization tool
- PASOE with database and web frontend
- Available in OpenEdge 12.8.9+ / 13.0+
- Analyzes .oemp files and provides visual tools to inspect the data
- Copy .oemp file in import folder of oemp PASOE instance. This will be analyzed to one level deep sub folders
- Open the web page (or refresh if already open)
- Overview shows the .oemp file ready to import. This will read the file into the OEMP database
- Review the profile report

Demo: Memory Profiler



Questions!

