OpenEdge Memory Profiler Recording Format

Last updated: April 2025

Overview

This document describes the format of the recordings generated by the AVM Memory Profiler. The recordings are presently written to disk in a file with a ".oemp" suffix.

The format is a proprietary format tailored to be read by an ABL Application (like the ABL CPU Profiler's output file).

A Memory Profiler recording is comprised of a series of memory use "snapshots", each providing a picture of memory use by the AVM at a point in time. Among other things, each snapshot has a sequence number, the UUID of the recording, and a timestamp; these enable the consumer of the recording to relate the different snapshots to one another and establish the order of the snapshots in the recording.

Each snapshot only reports what has changed since the previous snapshot; thus, the snapshots are incremental reports. For the Application Objects, it will only include what new objects have been created and what existing objects have been updated or deleted. The snapshot also includes new Call-Tree entries, new ABL Source names, and new "Other" names (see below).

Recording Sections and Format Overview

Each snapshot is comprised of a fixed number of sections – 11 in Version 1, 12 in Version 2.

Format Overview

Each piece of data in the snapshot will be either a number or a quoted string. Each section has a fixed number of columns for each line of data. Each column is delimited by a space.

Each section is terminated by a single period ('.') by itself on a line.

This format is conducive to being read into a database by an ABL program and will be familiar to ABL programmers.

For example, here you can see the first section, which is the header section and only ever has one line of data; line 2 is therefore the section delimiter; that's followed by the Platform Objects Catalog section, which has just two columns. See the column descriptions for each section below.

```
1 3277 1 "2022-02-01T08:01:39.967+05:00" "AS-4" "_AUTO" "?:?:?" ""...
.                       // Header section has just one line
"Memory Profiler" 1     // all data items for each element on its own line
"General" 2
   ...                  // all other Platform Object Catalog elements
.
   ...                  // all other sections
.                       // last section's terminator
```

As you will see, most of the data in the snapshots is numeric, and the data is highly relational.

Snapshot Sections Overview

Following are the snapshot sections in the order in which they're presented in the recording for Version 2 (V2) of the .oemp, which is the latest. For the ordering for Version 1 of the .oemp, see the Appendix. Details of the data items for each section will follow. The section names used below do not show up in the snapshot recordings themselves but are useful in this document for relating the data in one section to data in another.

1) Header – Holds info like the memory profiler recording version, process id, start time, snapshot sequence number, ABL Session ID (for PASOE), UUID for the memory profiling session (to relate all snapshots to one another).
2) PlatObjCatalog – A catalog of each Platform Object type and their unique integer identifiers. For example: RCode Directory (-D), Memory Profiler, Temp-table DB (-Bt), Runtime Stack (-s), Compiler, CPU Profiler, Record Buffers (-l), etc. The unique integer identifier corresponds to the items in the PlatObjData section. This is a static list for the life of the ABL Memory Profiling session, so this section is only reported in the first snapshot and thereafter will empty.
3) AppObj Catalog – A catalog of all the Application Object types and their unique integer identifiers. For example, Dynamic Query, OOABL User-defined, DB Connection, Dataset, UI Window, UI Button, Large Object, Widget Pool, etc. The unique integer identifier corresponds to the items in the AppObjData section. Like the PlatObjCatalog, this is a static list, but will grow as we add new Application

Object types to the AVM. This section is only reported in the first snapshot and is empty after that.

4) BIObjCatalog – A catalog of all the OOABL Built-in Classes and their unique integer identifiers. For example, Progress.Lang.Object, Progress.Lang.Class, Progress.Lang.Error, Progress.Json.JsonParser, Progress.Collections.List, etc. The unique integer identifier corresponds to the items in the AppObjData section. Like the other two catalogs, this is a static list, but will grow as we add more built-in Classes to the AVM. Like the PlatObjCatalog and the AppObj Catalog sections, this section is only reported with the first snapshot.
5) PlatObjData – This section has the memory use data for various AVM systems that support the application (the Platform Level Category). Each entry has an integer identifier ("Sid", below) that corresponds to one of the unique identifiers in the PlatObjCatalog section.
6) ABLNameList – This section is a list of the ABL sources that have been executed during the life of the ABL Memory Profiler Session. This data is like the module-id data in the ABL CPU Profiler. Each ABL source has an integer identifier ("SrcNameId") which is unique within the ABLNameList section; for entries in the AppObjData section that are of type "Procedure" or "OOABL Obj", the integer source identifier ("SrcId") from that section corresponds to the ABL source identifier in this section ("SrcNameId").
7) OtherNameList – The ABLNameList section describes the names of the ABL Application's procedure, classes, methods, etc. This section is a list of other Application Objects which have names – for example, database connections, named widget pools, dynamic temp-tables, etc.
8) CallTreeNodes – This section is a list of the nodes that comprise the call-tree which has been constructed over the life of the ABL Memory Profiler Session. This data is like the call-tree data from the ABL CPU Profiler. This data, together with the ABLNameList section, provides the means for the tool to present the call-stack at the point where any Application Object was created. Each entry in the AppObjData section has an integer node identifier ("AppObjNodeId", see below) that corresponds to the unique integer node identifier in this section ("NodeId"), identifying where in the call-tree the Application Object was created. See the example code mentioned in the Overview for how to construct the Call-Stack from the CallTreeNodes data.
9) AppObjData – This section has the memory use data and details for objects that are managed by the Application itself (the Application Object types). For example, Procedure instances, OOABL User-defined instances, MEMPTRs, Dynamic Queries, etc. Each entry has an integer type ("AppObjTyp", see below) that corresponds to

one of the unique identifiers in the AppObj Catalog section. It also has an unsigned 64-bit integer object identifier ("AppObjId", see below) that is unique among the Application Objects in the lifetime of this ABL Memory Profiling Session. In Version 2 of the .oemp file, this section only holds new objects which have not yet been recorded. All changed objects data goes into the next section (V2).

10) ChangedAppObjData – This section is new in Version 2 (V2). It holds all the changed objects data since the prior snapshot was taken. The main thing that changes once an Object is created is its size.
11) DeletedObjs – This section is a list of every Application Object that has been deleted since the previous snapshot was generated. The Application Objects are identified by their unsigned 64-bit integer object identifier ("AppObjId") which corresponds to the object identifier in the AppObjData section.
12) Trailer – This final section holds the timestamp of snapshot, some internal statistics, and status of the AVM Memory Profiler. It also holds totals for all Application Objects memory use ("appMemTot") and all Platform Objects memory use.

Details of Data per Section

Each of the following sub-sections describes the datatype, data item name, and meaning of each of the columns in the snapshot sections and how they relate to data items in the other sections. When referring to a data item in another section, we refer to it by *SectionName.DataItemName* notation. For example, Header.StartTime or AppObjData.Mem.

The order of the data in each section presented here is the order of the columns in the section.

1. Header Section
    a. Integer "Version" – Memory Profiler recording format version
    b. Integer "ProcessId" – Process id for this ABL Client or PASOE Agent process.
    c. Integer "SnapshotSeq" – Which snapshot is this in the series of snapshots recorded since ABL Memory Profiling began.
    d. String "StartTime" – UTC formatted date time string describing when the ABL Memory Profiler Session began.
    e. String "SessionId" – "AVM-Client" for ABL Client processes; ABL Session Id for PASOE (AS-4, AS-7, etc.).

f. String “Tag” – Special identifier for the type of snapshot; indicates whether the snapshot was generated automatically at a time interval, at the end of a PASOE request, or programmatically from within the Application via MemoryProfiler:TakeSnapshot().

g. String “RqId” – Nil for ABL Client processes; Request identifier for PASOE.

h. String “RecordingType” – Unused. Was in memory profiler output version 0.

i. Unsigned Integer “RequestStartTime” – Nil for ABL Client processes; Time the current request began running; Microseconds since start of ABL Memory Profiling Session (Header.StartTime).

j. String “UUID” – Unique identifier for this Memory Profiling session, relating the series of snapshots. This will be important once we support not just writing the recording to a file, but also allow writing the snapshots to a remote server URL, where the server may be processing snapshots from multiple sources simultaneously.

k. String “PROPATH” – PROPATH setting at the time the snapshot is written. However, this will be nil unless the PROPATH has changed since the last time the PROPATH was written in a snapshot Header.

l. 64-bit Integer “OverheadMem” – Zero for ABL Client processes; bytes of overhead memory used by the PASOE agent as reported in the AgentManager’s AgentStatHist.OverheadMemory.

m. String “AppName” – Application name that this PASOE agent is serving. For ABL Client processes: this is Nil for Version 1 of .oemp; for Version 2 of .oemp (V2) it is the name of the -p startup parameter.

n. String “RequestAPI” – Nil for ABL Client processes; Name of the top-level API called for this request.

o. Decimal “CPU” – Current Percent CPU utilization at the time of this snapshot, with precision 2. (e.g., 22.45, 97.02, 0.15)

p. String “OEVersion” – Current version of OpenEdge this .oemp was generated by. New in Version 2 of the .oemp (V2).

q. String “PASOEInstanceAlias” – Nil for ABL Clients, but the name of the PASOE instance for PASOE .oemp’s. New in Version 2 of the .oemp (V2).

r. String “Description” – This is taken from the “Description” property for the -profileMemory memprof.cfg file. Maximum size is 80 characters. New in Version 2 of the .oemp (V2).

2. PlatObjCatalog
    a. String “PlatObjName” – Name of the Platform Object category
    b. Integer “PlatObjId” – Unique identifier
3. AppObj Catalog
    a. String “AppObjTypeName” – Name of the Application Object Type
    b. Integer “AppObjTypeId” – Unique identifier
4. BIObjCatalog
    a. String “BIObjTypeName” – Name of the Built-in OOABL Class
    b. Integer “BIObjTypeId” – Unique identifier
5. PlatObjData
    a. Integer “PlatObjId” – Platform object identifier. Corresponds to PlatObjCatalog.PlatObjId.
    b. 64-bit Integer “Mem” – Memory used by this platform object.
    c. 64-bit Integer “Abk” – Internal use only. Memory allocated for blocks in pool.
    d. 64-bit Integer “PoolAllocs” – Internal use only. Memory used from blocks in pool.
6. ABLNameList
    a. Integer “SrcNameId” – Unique identifier within names in all of ABLNameList
    b. String “SrcName”– Name of the ABL procedure, class, method, function, etc.
    c. Integer “StartLine” – Line number where this method or function declaration begins; 0 for .p or .cls. Presently necessary for distinguishing between overloaded method names.
7. OtherNameList
    a. Integer “NameId” – Unique identifier within names in all of “OtherNameList”
    b. String “Name”– Name used in the Application
8. CallTreeNodes
    a. Integer “NodeId” – Node unique identifier within this Memory Profiling Session.
    b. Integer “SrcId” – ABL Source identifier. Corresponds to ABLNameList.SrcNameId

c. Integer "ParentNodeId" – Parent node identifier. Will be zero if this node is a root node. Corresponds to CallTreeNodes.NodeId
d. Integer "ParentLine" – Line number in parent node where this node was called. Will be zero if this node is the root node.

9. AppObjData – Note that the only Objects that will be recorded on in this section are both, a) those whose Application Object Type has been instrumented in the AVM (this will be completed over time), and b) those objects which were created while Memory Profiling was enabled. In the initial release, memory profiling may only be enabled at startup, so that means all Objects whose types are instrumented in the AVM will be in this section. However, once we support memory profiling enablement *after* startup, those objects that were created *before* enablement will not be recorded here.
   a. Unsigned 64-bit Integer "AppObjId" – Unique identifier within all Application Objects in this Memory Profiling Session.
   b. Integer "AppObjTyp" – Type of Application Object. Corresponds to AppObjCatalog.AppObjTypeId.
   c. 64-bit Integer "Mem" – Bytes of memory used by this Application Object.
   d. 64-bit Integer "Abk" – *Internal use only.* Bytes of memory allocated for blocks in pool.
   e. 64-bit Integer "PoolAllocs" – *Internal use only.* Bytes of memory used from blocks in pool.
   f. Unsigned Integer "CreationTime" – Time that this object was created. Microseconds since start of ABL Memory Profiling Session (Header.StartTime).
   g. Integer "AppObjNodeId" – Node where this object was created. Corresponds to CallTreeNodes.NodeId. The info in the Node itself contains the corresponding source Id for the .p/.cls which created this Application Object.
   h. Integer "SrcLine" – Line number in the Call-Tree node (AppObjData.AppObjNodeId) where this object was created. Note that the .p/.cls source where the Application Object was created is identified in the Node (AppObjData.AppObjNodeId).
   i. Integer "SrcId" – ABL source identifier. Corresponds to ABLNameList.SrcNameId. Note that this will be Zero for *all except* where AppObjData.AppObjTyp is Procedure, OOABL Obj, OOABL Static Obj, or Reusable Obj.
   j. Unsigned 64-bit Integer – Unused. Was in memory profiler output version 0.
   k. Unsigned 64-bit Integer "ScopedTo" – Application Object identifier (AppObjData.AppObjId) of object to which this object is scoped. Non-persistent Widget-Pools and defined temp-tables are scoped to the object (Application Object Type: Procedure, OOABL Obj, OOABL Static Obj, or Reusable Obj) in which they are defined. Dynamic Objects are often scoped to a Widget-Pool.

l. Integer "TempTableNum" – Temp-table table number in the temp-table database. Only set when this object is one of the temp-table Application Object types (AppObjData.AppObjTyp). This number corresponds to the table id in the temp-table database. If this value is Zero, and the AppObjTyp is one of the temp-table objects, then that means that the temp-table schema has been set up, but the temp-table itself hasn't yet been instantiated in the temp-table database. This may be useful information to our customers because it identifies which temp-tables have not been referenced.

m. Integer "NameId" – Name identifier for this object. Corresponds to OtherNameList.NameId. Not set for every type of Application Object since many are not assigned a name. Always set to Zero for Procedure, OOABL Obj, OOABL Static Obj, and Reusable Obj type objects since their names are represented in the ABLNameList section. This may be set for certain Built-in Types (see below, under BioTypId).

n. Integer – Unused. Was in memory profiler output version 0.

o. Integer "BioTypId" – Only valid when AppObjData.AppObjTyp is OO Builtin. This is the identifier for the built-in class this object is. Corresponds to BIObjCatalog.BIObjTypeId. Note that the name of the Built-in is typically the name in the BIObjCatalog. But for Built-in's that are generic, such as Progress.Collection.List, the name will be stored in AppObjData.NameId. For example, "Progress.Collection.List<dog>"

p. Unsigned 64-bit Integer – Unused. Was in memory profiler output version 0.

A Note About Obtaining an Application Object's Name

- The AppObjectData.NameId field will be 0 if there is no name for the Application object *or* if the object's type is OOABL Obj, Procedure, OOABL Static Obj, or Reusable Object (in the future, possibly also .NET Obj).
- Apart from those 4 types (and possibly .NET Obj), all objects that have a name will have a non-zero entry in that column; examples are DB Connection or Named Widget Pool types. Use the AppObjectData.NameId to look up the name in the OtherNameList.
- For those 4 types (and possibly .NET Obj), the name should be looked up in the ABLNameList using the AppObjectData.SrcId.

- Application objects of the OO Builtin type require special care. If the AppObjectData.NameId field is non-zero, then use the name from the OtherNameList (this is where, for example, “Progress.Collection.List<dog>” will show up). But if AppObjectData.NameId is 0, then get its name from BIObjCatalog for that AppObjectData.BioTypId.

10. ChangedAppObjData
    a. Unsigned 64-bit Integer “AppObjId” – Identifier for Application Object. Corresponds to AppObjData.AppObjId.
    b. Integer “AppObjTyp” – Type of Application Object. Corresponds to AppObjCatalog.AppObjTypeId.
    c. 64-bit Integer “Mem” – Bytes of memory used by this Application Object.
    d. 64-bit Integer “Abk” – *Internal use only.* Bytes of memory allocated for blocks in pool.
    e. 64-bit Integer “PoolAllocs” – *Internal use only.* Bytes of memory used from blocks in pool.
    f. Integer “TempTableNum” – Temp-table table number in the temp-table database. Only set when this object is one of the temp-table Application Object types (AppObjData.AppObjTyp). This number corresponds to the table id in the temp-table database. If this value is Zero, and the AppObjTyp is one of the temp-table objects, then that means that the temp-table schema has been set up, but the temp-table itself hasn’t yet been instantiated in the temp-table database. This may be useful information to our customers because it identifies which temp-tables have not been referenced.
11. DeletedObjs
    a. Unsigned 64-bit Integer “AppObjId” – Identifier for Application Object. Corresponds to AppObjData.AppObjId.
    b. Unsigned Integer “DelTime” – Time that this object was deleted. Microseconds since start of ABL Memory Profiling Session (Header.StartTime).
12. Trailer
    a. Unsigned Integer “RecordingTime” – Time of the recording, in microseconds since the Header.StartTime.
    b. Integer “AppObjCnt” – Number of active Application Objects
    c. 64-bit Integer “appMemTot” – Total memory used by Application Objects
    d. 64-bit Integer “platMemTot” – Total memory used by Platform Objects
    e. Integer “MaxAppObjs” – Maximum number of concurrent Application Objects
    f. Integer “MaxHier” – Maximum hierarchy of OOABL Class
    g. Integer – Unused. Was in memory profiler output version 0.

h. Integer “reuseObjCnt” – Total number of reusable objects (not included in the above “AppObjCnt”. New in Version 2 of the .oemp (V2).
i. 64-bit Integer “reuseMemTot” – Total memory used by reusable Application Objects (not included in “appMemTot”. New in Version 2 of the .oemp (V2).
j. String “Tag” – Same Header.Tag value that is stored in the Header. New in Version 2 of the .oemp (V2).
k. Integer “SnapshotSeq” – Same Header.SnapshotSeq value that is stored in the Header. “New in Version 2 of the .oemp (V2).

Other Notes about the Sections in the Snapshot

Recall that each snapshot is an incremental report; each snapshot of the memory profiler recording has data only for those Application Objects that have changed (been created, deleted, or updated) since the previous snapshot.

Almost all the sections of a recording described above are either static and fixed or they are static and growing. The only sections with data that is dynamic and changing are the AppObjData section, the ChangedAppObjData section (V2), and the DeletedObjs section.

Header and Trailer are fixed recording and summary data.

The three catalogs (PlatObj, AppObj, and BIObj) are static and fixed.

The amounts in the PlatObjData section changes, but the elements are a fixed number.

The data in the CallTreeNodes, ABLNameList, and OtherNameList never changes, but continuously grows. For example, once the info for call-tree node number 7 has been reported, there’s nothing new about it that will be reported later. A Snapshot from early in the profiling session is not disturbed by later additions to the CallTreeNodes, ABLNameList, or OtherNameList sections because that earlier snapshot won’t reference any of the later additions. The new data in these sections is published as soon as possible (i.e., in the next snapshot) after it first turns up in the Application.

Therefore, the AppObjData, ChangedAppObjData (new in V2), and DeletedObjs sections are the only ones that are subject to the incremental nature of snapshots.

Appendix – Version 1 .oemp Structure

The descriptions and the fields of each section are the same as in Version 2 of the .oemp, except where the description above notes that a field is a Version 2 (V2) field. In version 1,

the AppObjData had all Application object data: newly created Application objects *and* any changed Application objects. Version 2 (V2) separates those into their own sections and orders the sections differently. Following is the order of sections for version 1.

1) Header

2) PlatObjCatalog

3) AppObj Catalog

4) BIObjCatalog

5) AppObjData

6) PlatObjData

7) DeletedObjs

8) CallTreeNodes

9) ABLNameList

10) OtherNameList

11) Trailer