Java Memory Leaks

The garbage collection feature of the JVM greatly reduces the opportunities to introduce memory leaks into a codebase. However, because garbage collection does not eliminate memory leaks completely, AppDynamics includes Automatic Leak Detection for supported JVMs.

Permissions

To enable Automatic Leak Detection, you need the Permissions.

To start an On Demand Capture Session, you need the Configure Agent Properties permission.

For information on AppDynamics Role Based Access Control, see Roles and Permissions.

Overview of Automatic Leak Detection

You can access Automatic Leak Detection on the Memory tab of the Node Dashboard. Automatic Leak Detection is disabled by default because it increases overhead on the JVM. You should enable leak detection mode only when you suspect a memory leak problem. Turn off Automatic Leak Detection after you identify the cause for the leak.

Automatic Leak Detection uses On Demand Capture Sessions to capture actively used collections, any class that implements JDK Map or Collection interface during the capture period. The default capture period is 10 minutes.

AppDynamics tracks every Java collection that meets the following criteria:
• The collection has been alive for at least 30 minutes.
• The collection has at least 1000 elements.
• The collection Deep Size is at least 5 MB. The agent calculates Deep Size by traversing recursive object graphs of all the objects in the collection.

The following node properties define the defaults for leak detection criteria:

• minimum-age-for-evaluation-in-minutes
• minimum-number-of-elements-in-collection-to-deep-size
• minimum-size-for-evaluation-in-mb

See App Agent Node Properties for more information.

The Java Agent tracks the collection and identifies potential leaks using a linear regression model. You can identify the root cause of the leak by tracking frequent access to the collection over a period of time.

After it qualifies a collection, AppDynamics monitors the collection size for a long term growth trend. Positive growth indicates the collection is the potential source of a memory leak.

After AppDynamics identifies a leaking collection, the Java Agent automatically triggers diagnostics every 30 minutes. The diagnostics capture a shallow content dump and activity traces of the code path and business transactions that access the collection. You can drill down into any leaking collection monitored by the agent, to manually trigger Content Summary Capture and Access Tracking sessions.

You can also monitor memory leaks for custom memory structures. Typically custom memory structures are used as caching solutions. In a distributed environment, caching can easily become a prime source of memory leaks. It is therefore important to manage and track memory statistics for these memory structures. To do this, you must first configure custom memory structures. See Custom Memory Structures for Java.

Workflow to Troubleshoot Memory Leaks

Use the following workflow to troubleshoot memory leaks on JVMs that have been identified with a potential memory leak problem:

1. Monitor memory for potential JVM memory leaks
2. Enable automatic leak detection
3. Start on demand capture session
4. Detect and troubleshoot leaking condition

These steps are described in the following sections.

Monitor Memory for Potential JVM Leaks

Use the Node Dashboard to identify the memory leak. A possible memory leak is indicated by a growing trend in the heap as well as the old / tenured generation memory pool.

An object is automatically marked as a potentially leaking object when it shows a positive and steep growth slope.

The Automatic Memory Leak dashboard shows:

• Collection Size: The number of elements in a collection.
• Potentially Leaking: Potentially leaking collections are marked as red. You should start diagnostic sessions on potentially leaking objects.
• Status: Indicates if a diagnostic session has been started on an object.
• Collection Size Trend: A positive and steep growth slope indicates a potential memory leak.

# of Elements in a Collection

<table>
<thead>
<tr>
<th>Class</th>
<th>Collection Size</th>
<th>Potentially Leaking</th>
<th>Object Creation Time</th>
<th>JVM Start Time</th>
<th>Object Insta...</th>
<th>Status</th>
<th>Collection Size Trend</th>
</tr>
</thead>
</table>

Tip: To identify long-lived collections compare the JVM start time and Object Creation Time.

If you cannot see any captured collections, ensure that you have correct configuration for detecting potential memory leaks.
Enable Memory Leak Detection

Memory leak detection is available through the Automatic Leak Detection feature. Once the Automatic Leak Detection feature is turned on and a capture session has been started, AppDynamics tracks all frequently used collections. Therefore, using this mode results in higher overhead.

Turn on Automatic Leak Detection mode only when a memory leak problem is identified and then click **Start On Demand Capture Session** to start monitoring frequently used collections and detect leaking collections.

To achieve optimum performance, start diagnosis on one individual collection at a time.

Turn the capture session and the leak detections modes off after you have identified and resolved the leak.

Troubleshoot Memory Leaks

After detecting a potential memory leak, troubleshooting the leak involves performing the following three actions:

- Select the Collection Object that you want to monitor
- Use Content Inspection
- Use Access Tracking

Select the Collection Object to Monitor

On the Automatic Leak Detection dashboard, right-click the class name and click **Drill Down**.

For performance reasons, start the troubleshooting session on a single collection object at a time.

Use Content Inspection

Content Inspection identifies which part of the application the collection belongs to so that you can start troubleshooting. It allows monitoring histograms of all the elements in a particular collection.

Enable Automatic Leak Detection by starting an On Demand Capture Session, select the object you want to troubleshoot, and then follow the steps listed below:

1. Click the **Content Inspection** tab.
2. Click **Start Content Summary Capture Session** to start the content inspection session.
3. Enter the session duration. Allow at least 1 – 2 minutes for data generation.
4. Click **Refresh** to retrieve the session data.
5. Click on the snapshot to view details about an individual session.

Use Access Tracking

Use Access Tracking to view the actual code paths and business transactions accessing the collections object.
As described above in Workflow to Troubleshoot Memory Leaks, enable Automatic Leak Detection, start an On Demand Capture Session, select the object you want to troubleshoot, and then follow the steps listed below:

1. Select the Access Tracking tab
2. Click Start Access Tracking Session to start the tracking session.
3. Enter the session duration. Allow at least 1-2 minutes for data generation.
4. Click Refresh to retrieve session data.
5. Click on the snapshot to view details about an individual session.

The troubleshooting information pane shows the Java stack trace associated with the session. By default, the stack trace is shown to a depth of 10 lines. If you would like to temporarily increase the number of lines captures, you can use the maximum-activity-trace-stack-depth Java Agent property described on App Agent Node Properties Reference.

Increasing the stack trace depth can consume a significant amount of system resources. Be sure to remove the property or set it back to the default value of 10 after you have captured the desired information.