



# Garbage Collection in the BlackBerry Java Development Environment

## Contents

Garbage collection .....	2
RAM garbage collection .....	2
Full garbage collection.....	3
Persistent garbage collection .....	3
Emergency garbage collection.....	3
Thorough garbage collection .....	4
Idle garbage collection .....	4
Programmatic invocation of garbage collection .....	4
RAM panic.....	4

## Garbage collection

To understand proper memory management techniques, you must understand how the garbage collection system works on BlackBerry® Wireless Handhelds. The garbage collection system removes unnecessary objects at key times, which enables the system to function at an optimal level.

**Note:** When the system performs a garbage collection, all the system threads are paused until the garbage collection is complete.

Garbage collection types	Run time	Run conditions	Run effects
RAM garbage collection	~0.5 seconds	<ul style="list-style-type: none"> <li>RAM panic</li> </ul>	<ul style="list-style-type: none"> <li>removes unreferenced objects from RAM</li> </ul>
Full garbage collection	~ 1 seconds	<ul style="list-style-type: none"> <li>RAM panic</li> <li>process exceeds heap size</li> <li>lack of persistent object handles</li> <li><code>System.GC()</code> invocation</li> </ul>	<ul style="list-style-type: none"> <li>performs RAM garbage collection</li> <li>removes non-persistent unreferenced objects from flash memory</li> <li>marks unreferenced persistent objects</li> <li>releases non-persistent RAM and object handles</li> </ul>
Persistent garbage collection	10-15 seconds	<ul style="list-style-type: none"> <li>lack of persistent object handles</li> <li>low resources during a commit operation</li> <li><code>System.GC()</code> invocation</li> </ul>	<ul style="list-style-type: none"> <li>performs full garbage collection</li> <li>performs auto-commit</li> <li>releases unreferenced persistent objects</li> <li>releases persistent object handles</li> </ul>
Emergency garbage collection	20 seconds	<ul style="list-style-type: none"> <li>low resources</li> <li>low available flash memory</li> <li>lack of object handles</li> <li>lack of persistent object handles</li> </ul>	<ul style="list-style-type: none"> <li>performs persistent garbage collection</li> <li>recompacts flash records</li> </ul>
Thorough garbage collection	25-30 seconds	<ul style="list-style-type: none"> <li>handheld idle</li> </ul>	<ul style="list-style-type: none"> <li>performs emergency garbage collection</li> <li>pages RAM out to flash memory</li> </ul>

## RAM garbage collection

A RAM garbage collection removes unreferenced objects from RAM. A RAM garbage collection can only be initiated during a RAM panic operation when the VM cannot allocate an object because of a lack of space in RAM. See RAM panic on page 4 for more information.

Once initiated, the RAM garbage collection typically takes 500 to 600 milliseconds to execute. The garbage collection removes any freshly allocated variables that are no longer referenced in RAM. A RAM garbage collection can only be performed when objects have not been paged out to flash memory to make sure that a lack of a reference in RAM is a sufficient condition for removing the object.

---

## Full garbage collection

A full garbage collection might be initiated by the system in the following situations:

- RAM panic
- process is about to exceed its currently allocated heap size
- VM cannot allocate a new object because no object handles are available
- handheld is idle

The full garbage collection executes for 1 second on average and should take less than 2 seconds. The full garbage collection performs the following steps:

1. It performs a RAM garbage collection.
2. It marks unreferenced objects that are currently stored in flash memory but are not persisted.
3. It releases any non-persistent object handles in both RAM and flash memory.

## Persistent garbage collection

A persistent garbage collection attempts to remove unreferenced objects from persistent stores in addition to the objects that would be removed in the full garbage collection operation. The persistent garbage collection operation is only invoked in the following situations:

- The VM cannot allocate a new object because there are no available object handles and a full garbage collection did not free any object handles.
- During a commit operation, if the VM encounters low resources.
- In version 4.0 of the BlackBerry Handheld Software, invoking `System.GC()` initiates a persistent garbage collection as long as there are sufficient persistent objects that can be removed.

The persistent garbage collection operation typically runs for 10 to 15 seconds. During a persistent garbage collection operation, the system completes the following steps:

1. It performs an auto-commit. An auto-commit commits all the contents in persistent stores at the time of the garbage collection. An auto-commit prevents data loss during persistent garbage collection.
2. It performs a full garbage collection.
3. It removes the marked objects from the Full garbage collection. Unreferenced persistent object handles are also removed.

After a persistent garbage collection, both RAM and flash memory do not contain unreferenced objects.

## Emergency garbage collection

An emergency garbage collection occurs when the system starts to run out of necessary resources. The system monitors the level of the following resources:

- available flash memory
- object handles
- persistent object handles

If any of these items falls below a threshold, an emergency garbage collection is initiated. Once initiated, the emergency garbage collection runs for approximately 20 seconds. See *Low Memory Manager in the BlackBerry Java Development Environment* for more information.

During an emergency garbage collection, the system completes the following steps:

1. It performs a persistent garbage collection, including an auto-commit.

2. It recomacts flash memory records to maximize the amount of free space on the handheld.

### Thorough garbage collection

A thorough garbage collection is the most intensive garbage collection operation in the system. It is designed to run when the handheld is idle. A handheld is considered idle when the user has not pressed the keys or the trackwheel for a specific period of time. After a substantial amount of idle time, a thorough garbage collection can be initiated if it is required based on heuristics.

During a thorough garbage collection, the system completes the following steps:

1. It performs an emergency garbage collection.
2. It pages out RAM objects to flash memory to free RAM resources.

**Note:** If the handheld is used during a thorough garbage collection, the garbage collection automatically terminates, allowing the user full access to the handheld resources.

### Idle garbage collection

The system attempts to perform the following garbage collection operations when the handheld is idle. This enables the system to improve future system performance without impacting the user experience.

- Full garbage collection is performed when the handheld has been idle for a relatively small amount of time if required based on heuristics.
- Thorough garbage collection is performed when the handheld has been idle for a significant period of time if required based on heuristics.

**Note:** A garbage collection is not performed every time that the handheld is idle. It is only performed when the system considers garbage collection to be beneficial based on heuristics. This allows for optimal system performance and maximized battery performance.

### Programmatic invocation of garbage collection

Handheld software version	Description
Prior to version 3.8	Invoking <code>System.GC()</code> in handheld software versions prior to version 3.8 initiates a full garbage collection operation as described above.
Version 3.8 or later	The <code>System.GC()</code> method in version 3.8 of the handheld software has been modified to enable developers to invoke a persistent garbage collection if one is required by the system. During the <code>System.GC()</code> call, a full garbage collection is executed as before. However, at the end of the full garbage collection, the system uses heuristics to determine if a persistent garbage collection is required. If required, a persistent garbage collection is run.

### RAM panic

A RAM panic occurs when the VM runs out of available RAM. A RAM panic is initiated to recover RAM space. After each stage in the RAM panic, the VM attempts to allocate space for the object. If the VM succeeds, the RAM panic stops.

During RAM panic, the system completes one or more of the following steps:

1. It performs a RAM garbage collection.
2. It performs a full garbage collection.
3. It writes out old objects from RAM to flash memory.

- 
4. It pages the contents of RAM out to flash memory until flash memory is full or too much RAM has been paged out.

Part number: SWD\_X\_RIM(EN)-003.000

\*Check with service provider for availability, roaming arrangements and service plans. Certain features outlined in this document require a minimum version of BlackBerry Enterprise Server software, BlackBerry Desktop Software, and/or BlackBerry handheld software. May require additional application development. Prior to subscribing to or implementing any third party products or services, it is your responsibility to ensure that the airtime service provider you are working with has agreed to support all of the features of the third party products and services. Installation and use of third party products and services with RIM's products and services may require one or more patent, trademark or copyright licenses in order to avoid infringement of the intellectual property rights of others. You are solely responsible for determining whether such third party licenses are required and are responsible for acquiring any such licenses. To the extent that such intellectual property licenses may be required, RIM expressly recommends that you do not install or use these products and services until all such applicable licenses have been acquired by you or on your behalf. Your use of third party software shall be governed by and subject to you agreeing to the terms of separate software licenses, if any, for those products or services. Any third party products or services that are provided with RIM's products and services are provided "as is". RIM makes no representation, warranty or guarantee whatsoever in relation to the third party products and services and RIM assumes no liability whatsoever in relation to the third party products and services even if RIM has been advised of the possibility of such damages or can anticipate such damages.

© 2005 Research In Motion Limited. All rights reserved. The BlackBerry and RIM families of related marks, images and symbols are the exclusive properties of Research In Motion Limited. RIM, Research In Motion, BlackBerry and 'Always On, Always Connected' are registered with the U.S. Patent and Trademark Office and may be pending or registered in other countries.

All other brands, product names, company names, trademarks and service marks are the properties of their respective owners.

The handheld and/or associated software are protected by copyright, international treaties and various patents, including one or more of the following U.S. patents: 6,278,442; 6,271,605; 6,219,694; 6,075,470; 6,073,318; D,445,428; D,433,460; D,416,256. Other patents are registered or pending in various countries around the world. Please visit [www.rim.net/patents.shtml](http://www.rim.net/patents.shtml) for a current listing of applicable patents.

This document is provided "as is" and Research In Motion Limited (RIM) assumes no responsibility for any typographical, technical or other inaccuracies in this document. RIM reserves the right to periodically change information that is contained in this document; however, RIM makes no commitment to provide any such changes, updates, enhancements or other additions to this document to you in a timely manner or at all. RIM MAKES NO REPRESENTATIONS, WARRANTIES, CONDITIONS OR COVENANTS, EITHER EXPRESS OR IMPLIED (INCLUDING WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OF FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, MERCHANTABILITY, DURABILITY, TITLE, OR RELATED TO THE PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE REFERENCED HEREIN OR PERFORMANCE OF ANY SERVICES REFERENCED HEREIN). IN CONNECTION WITH YOUR USE OF THIS DOCUMENTATION, NEITHER RIM NOR ITS AFFILIATED COMPANIES AND THEIR RESPECTIVE DIRECTORS, OFFICERS, EMPLOYEES OR CONSULTANTS SHALL BE LIABLE TO YOU FOR ANY DAMAGES WHATSOEVER BE THEY DIRECT, ECONOMIC, COMMERCIAL, SPECIAL, CONSEQUENTIAL, INCIDENTAL, EXEMPLARY OR INDIRECT DAMAGES, EVEN IF RIM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, INCLUDING WITHOUT LIMITATION, LOSS OF BUSINESS REVENUE OR EARNINGS, LOST DATA, DAMAGES CAUSED BY DELAYS, LOST PROFITS, OR A FAILURE TO REALIZE EXPECTED SAVINGS.

This document might contain references to third party sources of information and/or third party web sites ("Third-Party Information"). RIM does not control, and is not responsible for, any Third-Party Information, including, without limitation the content, accuracy, copyright compliance, legality, decency, links, or any other aspect of Third-Party Information. The inclusion of Third-Party Information in this document does not imply endorsement by RIM of the third party in any way. Any dealings with third parties, including, without limitation, compliance with applicable licenses and terms and conditions, are solely between you and the third party. RIM shall not be responsible or liable for any part of such dealings.