



# Low Memory Manager in the BlackBerry Java Development Environment

## Contents

Low memory manager.....	1
Low memory manager conditions.....	1
BlackBerry handheld resources.....	1
Working with the low memory manager.....	1
Register your application as a LowMemoryListener.....	1
Handle events received by the LowMemoryListener.....	1
Free objects.....	2
Code sample.....	2

## Low memory manager

The low memory manager handles memory resources on the BlackBerry® Wireless Handheld when available memory resources fall below a certain threshold. The low memory manager attempts to free existing memory to provide more available memory on the handheld. All applications, including third-party applications, should work with the low memory manager to free as much memory as possible when the handheld is low on memory resources.

## Low memory manager conditions

The following conditions can cause the low memory manager to free memory resources:

- The amount of available flash memory on the handheld falls below a certain threshold. The free flash memory threshold depends on the amount of free RAM in the system. The free flash threshold varies between 400 KB and 800 KB.
- The number of persistent object handles that are available on the handheld falls below 1,000 handles.
- The number of object handles available on the handheld falls below a certain threshold. On current handhelds, this threshold is 1,000 object handles.

## BlackBerry handheld resources

Flash memory	Persistent object handles	Object handles
8 MB	12,000	24,000
16 MB	27,000	56,000
32 MB	65,000	132,000

## Working with the low memory manager

To take advantage of the low memory manager, applications complete the following steps:

1. Register as a `LowMemoryListener`.
2. Handle events received by the `LowMemoryListener`.

## Register your application as a `LowMemoryListener`

To use the low memory manager, register your application with the low memory manager. Provide an implementation of the `LowMemoryListener` interface in your application. When your application starts for the first time, register the `LowMemoryListener` implementation with the low memory manager.

**Note:** Do not register your listener more than once. If you register your listener more than once, you receive multiple calls when the low memory manager is invoked.

## Handle events received by the `LowMemoryListener`

To handle events received by the `LowMemoryListener`, implement the following method of the `LowMemoryListener` interface:

```
public boolean freeStaleObject( int priority )
```

This method is invoked when the low memory manager recognizes that the system is low on memory. It might also be invoked directly by a call to `LowMemoryManager.Poll()`.

### Handle events according to their priority

Your implementation of `freeStaleObject()` must take into account the priority of the event. There are three levels of priority: low, medium and high. In each case, the operation of the application could differ in terms of the memory resources it releases.

For example, when `freeStaleObject()` is invoked with low priority, the application should consider releasing transitory variables and any variables that are currently not necessary for complete functionality such as cached data.

When `freeStaleObject()` is invoked with medium priority, the application should consider removing stale data, such as very old email messages or old calendar appointments.

When `freeStaleObject()` is invoked with high priority, the application should remove objects in the application on a Least Recently Used basis. For example, when the low memory manager is invoked on the email application, it deletes messages at the bottom of the list of emails because those are likely to be the least used. The application should remove all its stale objects to reduce the amount of memory consumed on the handheld.

The `freeStaleObject()` method returns a boolean that indicates whether persistent data was released during the call to `freeStaleObject()`. If persistent data is released, return true. Return false otherwise.

**Note:** In practise, the low memory manager seldom specifies a priority higher than low priority.

## Free objects

To free persistent objects, applications invoke `LowMemoryManager.markAsRecoverable()` inside the implementations of `freeStaleObject()`.

Before invoking this method, applications should remove all references to the object and delete it from its data structures. The application then calls `LowMemoryManager.markAsRecoverable()`, passing in the object to free. This command indicates to the underlying JVM that the object can be removed as part of the low memory management operation.

You must invoke `markAsRecoverable()` because the low memory manager has a specific amount of memory that it is trying to recover. It distributes this amount across the registered applications. If your application frees memory resources, invoking `markAsRecoverable()` notifies the low memory manager of this fact so that it can count the freed memory against its target amount.

Implementations of `freeStaleObject()` should perform `commit()` operations when they delete data from their collections. Do not skip this step for efficiency because the low memory manager automatically uses a transactional commit operation so that all commits are deferred until the low memory manager operation completes. This enables you to use the same commit logic for low memory manager handlers and normal code.

## Code sample

The following is a sample implementation of `freeStaleObject()` that frees items from different persisted vectors associated with each of the priority levels.

```
/**
 * Sample implementation of LowMemoryListener.freeStaleObject().
 * This method is invoked when the low memory manager is
 * running out of memory and related resources. The application is asked to
 * free resources according to the priority level passed into this method.
 * @param priority The priority of the call, which is either High, Medium or
 * Low.
 * @return A boolean that indicates whether memory was freed by this call.
 * <strong>Note:</strong> the proper value must be returned from this method.
 */
```

```
public boolean freeStaleObject( int priority ) {
    boolean dataFreed = false;
    switch( priority ) {
        case LowMemoryListener.HIGH_PRIORITY:
            dataFreed = freeVector( _data._high );
            _priority = LowMemoryListener.LOW_PRIORITY;
            break;
        case LowMemoryListener.MEDIUM_PRIORITY:
            dataFreed = freeVector( _data._medium );
            _priority = LowMemoryListener.HIGH_PRIORITY;
            break;
        case LowMemoryListener.LOW_PRIORITY:
            dataFreed = freeVector( _data._low );
            _priority = LowMemoryListener.MEDIUM_PRIORITY;
            break;
    }
    if( dataFreed ) {
        _persist.commit();
    }
    return dataFreed;
}

/**
 * A private method that frees the priority vector.
 * @param vector The vector to free.
 * @return A boolean that indicates whether any objects were freed by this
 * method.
 */
private boolean freeVector( Vector vector ) {
    boolean dataFreed = false;
    int size = vector.size();
    for( int i = size - 1; i >= 0; i-- ) {
        Object obj = vector.elementAt( i );
        vector.removeElementAt( i );
        LowMemoryManager.markAsRecoverable( obj );
        dataFreed = true;
    }
    return dataFreed;
}
```

Part number: SWD\_X\_RIM(EN)-004.000

\*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.