| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageWriteParam.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/ImageWriter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/ImageWriteParam.html)    [**NO FRAMES**](http://docs.google.com/ImageWriteParam.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#147n2zr) | [METHOD](#ihv636) |

## **javax.imageio**

Class ImageWriteParam

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.imageio.IIOParam](http://docs.google.com/javax/imageio/IIOParam.html)  
 **javax.imageio.ImageWriteParam**

**Direct Known Subclasses:** [BMPImageWriteParam](http://docs.google.com/javax/imageio/plugins/bmp/BMPImageWriteParam.html), [JPEGImageWriteParam](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html)

public class **ImageWriteParam**extends [IIOParam](http://docs.google.com/javax/imageio/IIOParam.html)

A class describing how a stream is to be encoded. Instances of this class or its subclasses are used to supply prescriptive "how-to" information to instances of ImageWriter.

A plug-in for a specific image format may define a subclass of this class, and return objects of that class from the getDefaultWriteParam method of its ImageWriter implementation. For example, the built-in JPEG writer plug-in will return instances of javax.imageio.plugins.jpeg.JPEGImageWriteParam.

The region of the image to be written is determined by first intersecting the actual bounds of the image with the rectangle specified by IIOParam.setSourceRegion, if any. If the resulting rectangle has a width or height of zero, the writer will throw an IIOException. If the intersection is non-empty, writing will commence with the first subsampled pixel and include additional pixels within the intersected bounds according to the horizontal and vertical subsampling factors specified by [IIOParam.setSourceSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#setSourceSubsampling(int,%20int,%20int,%20int)).

Individual features such as tiling, progressive encoding, and compression may be set in one of four modes. MODE\_DISABLED disables the features; MODE\_DEFAULT enables the feature with writer-controlled parameter values; MODE\_EXPLICIT enables the feature and allows the use of a set method to provide additional parameters; and MODE\_COPY\_FROM\_METADATA copies relevant parameter values from the stream and image metadata objects passed to the writer. The default for all features is MODE\_COPY\_FROM\_METADATA. Non-standard features supplied in subclasses are encouraged, but not required to use a similar scheme.

Plug-in writers may extend the functionality of ImageWriteParam by providing a subclass that implements additional, plug-in specific interfaces. It is up to the plug-in to document what interfaces are available and how they are to be used. Writers will silently ignore any extended features of an ImageWriteParam subclass of which they are not aware. Also, they may ignore any optional features that they normally disable when creating their own ImageWriteParam instances via getDefaultWriteParam.

Note that unless a query method exists for a capability, it must be supported by all ImageWriter implementations (*e.g.* progressive encoding is optional, but subsampling must be supported).

**See Also:**[ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html)

| **Field Summary** | |
| --- | --- |
| protected  boolean | [**canOffsetTiles**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canOffsetTiles)            A boolean that is true if this ImageWriteParam allows tiling grid offset parameters to be set. |
| protected  boolean | [**canWriteCompressed**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteCompressed)            A boolean that is true if this writer can write images using compression. |
| protected  boolean | [**canWriteProgressive**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteProgressive)            A boolean that is true if this ImageWriteParam allows images to be written as a progressive sequence of increasing quality passes. |
| protected  boolean | [**canWriteTiles**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteTiles)            A boolean that is true if this ImageWriteParam allows tile width and tile height parameters to be set. |
| protected  int | [**compressionMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionMode)            The mode controlling compression settings, which must be set to one of the four MODE\_\* values. |
| protected  float | [**compressionQuality**](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionQuality)            A float containing the current compression quality setting. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**compressionType**](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionType)            A String containing the name of the current compression type, or null if none is set. |
| protected  [String](http://docs.google.com/java/lang/String.html)[] | [**compressionTypes**](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionTypes)            An array of Strings containing the names of the available compression types. |
| protected  [Locale](http://docs.google.com/java/util/Locale.html) | [**locale**](http://docs.google.com/javax/imageio/ImageWriteParam.html#locale)            A Locale to be used to localize compression type names and quality descriptions, or null to use a default Locale. |
| static int | [**MODE\_COPY\_FROM\_METADATA**](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA)            A constant value that may be passed into methods such as setTilingMode, setProgressiveMode, or setCompressionMode to enable that feature for future writes. |
| static int | [**MODE\_DEFAULT**](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT)            A constant value that may be passed into methods such as setTilingMode, setProgressiveMode, and setCompressionMode to enable that feature for future writes. |
| static int | [**MODE\_DISABLED**](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED)            A constant value that may be passed into methods such as setTilingMode, setProgressiveMode, and setCompressionMode to disable a feature for future writes. |
| static int | [**MODE\_EXPLICIT**](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT)            A constant value that may be passed into methods such as setTilingMode or setCompressionMode to enable a feature for future writes. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html)[] | [**preferredTileSizes**](http://docs.google.com/javax/imageio/ImageWriteParam.html#preferredTileSizes)            An array of preferred tile size range pairs. |
| protected  int | [**progressiveMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#progressiveMode)            The mode controlling progressive encoding, which must be set to one of the four MODE\_\* values, except MODE\_EXPLICIT. |
| protected  int | [**tileGridXOffset**](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileGridXOffset)            The amount by which the tile grid origin should be offset horizontally from the image origin if tiling has been set, or 0 otherwise. |
| protected  int | [**tileGridYOffset**](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileGridYOffset)            The amount by which the tile grid origin should be offset vertically from the image origin if tiling has been set, or 0 otherwise. |
| protected  int | [**tileHeight**](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileHeight)            The height of each tile if tiling has been set, or 0 otherwise. |
| protected  int | [**tileWidth**](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileWidth)            The width of each tile if tiling has been set, or 0 otherwise. |
| protected  int | [**tilingMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#tilingMode)            The mode controlling tiling settings, which Must be set to one of the four MODE\_\* values. |
| protected  boolean | [**tilingSet**](http://docs.google.com/javax/imageio/ImageWriteParam.html#tilingSet)            A boolean that is true if tiling parameters have been specified. |

| **Fields inherited from class javax.imageio.**[**IIOParam**](http://docs.google.com/javax/imageio/IIOParam.html) |
| --- |
| [controller](http://docs.google.com/javax/imageio/IIOParam.html#controller), [defaultController](http://docs.google.com/javax/imageio/IIOParam.html#defaultController), [destinationOffset](http://docs.google.com/javax/imageio/IIOParam.html#destinationOffset), [destinationType](http://docs.google.com/javax/imageio/IIOParam.html#destinationType), [sourceBands](http://docs.google.com/javax/imageio/IIOParam.html#sourceBands), [sourceRegion](http://docs.google.com/javax/imageio/IIOParam.html#sourceRegion), [sourceXSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#sourceXSubsampling), [sourceYSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#sourceYSubsampling), [subsamplingXOffset](http://docs.google.com/javax/imageio/IIOParam.html#subsamplingXOffset), [subsamplingYOffset](http://docs.google.com/javax/imageio/IIOParam.html#subsamplingYOffset) |

| **Constructor Summary** | |
| --- | --- |
| protected | [**ImageWriteParam**](http://docs.google.com/javax/imageio/ImageWriteParam.html#ImageWriteParam())()            Constructs an empty ImageWriteParam. |
|  | [**ImageWriteParam**](http://docs.google.com/javax/imageio/ImageWriteParam.html#ImageWriteParam(java.util.Locale))([Locale](http://docs.google.com/java/util/Locale.html) locale)            Constructs an ImageWriteParam set to use a given Locale. |

| **Method Summary** | |
| --- | --- |
| boolean | [**canOffsetTiles**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canOffsetTiles())()            Returns true if the writer can perform tiling with non-zero grid offsets while writing. |
| boolean | [**canWriteCompressed**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteCompressed())()            Returns true if this writer supports compression. |
| boolean | [**canWriteProgressive**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteProgressive())()            Returns true if the writer can write out images as a series of passes of progressively increasing quality. |
| boolean | [**canWriteTiles**](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteTiles())()            Returns true if the writer can perform tiling while writing. |
| float | [**getBitRate**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getBitRate(float))(float quality)            Returns a float indicating an estimate of the number of bits of output data for each bit of input image data at the given quality level. |
| int | [**getCompressionMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode())()            Returns the current compression mode, if compression is supported. |
| float | [**getCompressionQuality**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQuality())()            Returns the current compression quality setting. |
| [String](http://docs.google.com/java/lang/String.html)[] | [**getCompressionQualityDescriptions**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions())()            Returns an array of Strings that may be used along with getCompressionQualityValues as part of a user interface for setting or displaying the compression quality level. |
| float[] | [**getCompressionQualityValues**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityValues())()            Returns an array of floats that may be used along with getCompressionQualityDescriptions as part of a user interface for setting or displaying the compression quality level. |
| [String](http://docs.google.com/java/lang/String.html) | [**getCompressionType**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionType())()            Returns the currently set compression type, or null if none has been set. |
| [String](http://docs.google.com/java/lang/String.html)[] | [**getCompressionTypes**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionTypes())()            Returns a list of available compression types, as an array or Strings, or null if a compression type may not be chosen using these interfaces. |
| [Locale](http://docs.google.com/java/util/Locale.html) | [**getLocale**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getLocale())()            Returns the currently set Locale, or null if only a default Locale is supported. |
| [String](http://docs.google.com/java/lang/String.html) | [**getLocalizedCompressionTypeName**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getLocalizedCompressionTypeName())()            Returns a localized version of the name of the current compression type, using the Locale returned by getLocale. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html)[] | [**getPreferredTileSizes**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getPreferredTileSizes())()            Returns an array of Dimensions indicating the legal size ranges for tiles as they will be encoded in the output file or stream. |
| int | [**getProgressiveMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode())()            Returns the current mode for writing the stream in a progressive manner. |
| int | [**getTileGridXOffset**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridXOffset())()            Returns the horizontal tile grid offset of an image as it will be written to the output stream. |
| int | [**getTileGridYOffset**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridYOffset())()            Returns the vertical tile grid offset of an image as it will be written to the output stream. |
| int | [**getTileHeight**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileHeight())()            Returns the height of each tile in an image as it will be written to the output stream. |
| int | [**getTileWidth**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileWidth())()            Returns the width of each tile in an image as it will be written to the output stream. |
| int | [**getTilingMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode())()            Returns the current tiling mode, if tiling is supported. |
| boolean | [**isCompressionLossless**](http://docs.google.com/javax/imageio/ImageWriteParam.html#isCompressionLossless())()            Returns true if the current compression type provides lossless compression. |
| void | [**setCompressionMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int))(int mode)            Specifies whether compression is to be performed, and if so how compression parameters are to be determined. |
| void | [**setCompressionQuality**](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionQuality(float))(float quality)            Sets the compression quality to a value between 0 and 1. |
| void | [**setCompressionType**](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionType(java.lang.String))([String](http://docs.google.com/java/lang/String.html) compressionType)            Sets the compression type to one of the values indicated by getCompressionTypes. |
| void | [**setProgressiveMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int))(int mode)            Specifies that the writer is to write the image out in a progressive mode such that the stream will contain a series of scans of increasing quality. |
| void | [**setTiling**](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int))(int tileWidth, int tileHeight, int tileGridXOffset, int tileGridYOffset)            Specifies that the image should be tiled in the output stream. |
| void | [**setTilingMode**](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int))(int mode)            Determines whether the image will be tiled in the output stream and, if it will, how the tiling parameters will be determined. |
| void | [**unsetCompression**](http://docs.google.com/javax/imageio/ImageWriteParam.html#unsetCompression())()            Removes any previous compression type and quality settings. |
| void | [**unsetTiling**](http://docs.google.com/javax/imageio/ImageWriteParam.html#unsetTiling())()            Removes any previous tile grid parameters specified by calls to setTiling. |

| **Methods inherited from class javax.imageio.**[**IIOParam**](http://docs.google.com/javax/imageio/IIOParam.html) |
| --- |
| [activateController](http://docs.google.com/javax/imageio/IIOParam.html#activateController()), [getController](http://docs.google.com/javax/imageio/IIOParam.html#getController()), [getDefaultController](http://docs.google.com/javax/imageio/IIOParam.html#getDefaultController()), [getDestinationOffset](http://docs.google.com/javax/imageio/IIOParam.html#getDestinationOffset()), [getDestinationType](http://docs.google.com/javax/imageio/IIOParam.html#getDestinationType()), [getSourceBands](http://docs.google.com/javax/imageio/IIOParam.html#getSourceBands()), [getSourceRegion](http://docs.google.com/javax/imageio/IIOParam.html#getSourceRegion()), [getSourceXSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#getSourceXSubsampling()), [getSourceYSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#getSourceYSubsampling()), [getSubsamplingXOffset](http://docs.google.com/javax/imageio/IIOParam.html#getSubsamplingXOffset()), [getSubsamplingYOffset](http://docs.google.com/javax/imageio/IIOParam.html#getSubsamplingYOffset()), [hasController](http://docs.google.com/javax/imageio/IIOParam.html#hasController()), [setController](http://docs.google.com/javax/imageio/IIOParam.html#setController(javax.imageio.IIOParamController)), [setDestinationOffset](http://docs.google.com/javax/imageio/IIOParam.html#setDestinationOffset(java.awt.Point)), [setDestinationType](http://docs.google.com/javax/imageio/IIOParam.html#setDestinationType(javax.imageio.ImageTypeSpecifier)), [setSourceBands](http://docs.google.com/javax/imageio/IIOParam.html#setSourceBands(int%5B%5D)), [setSourceRegion](http://docs.google.com/javax/imageio/IIOParam.html#setSourceRegion(java.awt.Rectangle)), [setSourceSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#setSourceSubsampling(int,%20int,%20int,%20int)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### MODE\_DISABLED

public static final int **MODE\_DISABLED**

A constant value that may be passed into methods such as setTilingMode, setProgressiveMode, and setCompressionMode to disable a feature for future writes. That is, when this mode is set the stream will **not** be tiled, progressive, or compressed, and the relevant accessor methods will throw an IllegalStateException.

**See Also:**[MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode()), [setTilingMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int)), [getTilingMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode()), [setCompressionMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int)), [getCompressionMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.imageio.ImageWriteParam.MODE_DISABLED)

### MODE\_DEFAULT

public static final int **MODE\_DEFAULT**

A constant value that may be passed into methods such as setTilingMode, setProgressiveMode, and setCompressionMode to enable that feature for future writes. That is, when this mode is enabled the stream will be tiled, progressive, or compressed according to a sensible default chosen internally by the writer in a plug-in dependent way, and the relevant accessor methods will throw an IllegalStateException.

**See Also:**[MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode()), [setTilingMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int)), [getTilingMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode()), [setCompressionMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int)), [getCompressionMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.imageio.ImageWriteParam.MODE_DEFAULT)

### MODE\_EXPLICIT

public static final int **MODE\_EXPLICIT**

A constant value that may be passed into methods such as setTilingMode or setCompressionMode to enable a feature for future writes. That is, when this mode is set the stream will be tiled or compressed according to additional information supplied to the corresponding set methods in this class and retrievable from the corresponding get methods. Note that this mode is not supported for progressive output.

**See Also:**[MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode()), [setTilingMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int)), [getTilingMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode()), [setCompressionMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int)), [getCompressionMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.imageio.ImageWriteParam.MODE_EXPLICIT)

### MODE\_COPY\_FROM\_METADATA

public static final int **MODE\_COPY\_FROM\_METADATA**

A constant value that may be passed into methods such as setTilingMode, setProgressiveMode, or setCompressionMode to enable that feature for future writes. That is, when this mode is enabled the stream will be tiled, progressive, or compressed based on the contents of stream and/or image metadata passed into the write operation, and any relevant accessor methods will throw an IllegalStateException.

This is the default mode for all features, so that a read including metadata followed by a write including metadata will preserve as much information as possible.

**See Also:**[MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode()), [setTilingMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int)), [getTilingMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode()), [setCompressionMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int)), [getCompressionMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.imageio.ImageWriteParam.MODE_COPY_FROM_METADATA)

### canWriteTiles

protected boolean **canWriteTiles**

A boolean that is true if this ImageWriteParam allows tile width and tile height parameters to be set. By default, the value is false. Subclasses must set the value manually.

Subclasses that do not support writing tiles should ensure that this value is set to false.

### tilingMode

protected int **tilingMode**

The mode controlling tiling settings, which Must be set to one of the four MODE\_\* values. The default is MODE\_COPY\_FROM\_METADATA.

Subclasses that do not writing tiles may ignore this value.

**See Also:**[MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [setTilingMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int)), [getTilingMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode())

### preferredTileSizes

protected [Dimension](http://docs.google.com/java/awt/Dimension.html)[] **preferredTileSizes**

An array of preferred tile size range pairs. The default value is null, which indicates that there are no preferred sizes. If the value is non-null, it must have an even length of at least two.

Subclasses that do not support writing tiles may ignore this value.

**See Also:**[getPreferredTileSizes()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getPreferredTileSizes())

### tilingSet

protected boolean **tilingSet**

A boolean that is true if tiling parameters have been specified.

Subclasses that do not support writing tiles may ignore this value.

### tileWidth

protected int **tileWidth**

The width of each tile if tiling has been set, or 0 otherwise.

Subclasses that do not support tiling may ignore this value.

### tileHeight

protected int **tileHeight**

The height of each tile if tiling has been set, or 0 otherwise. The initial value is 0.

Subclasses that do not support tiling may ignore this value.

### canOffsetTiles

protected boolean **canOffsetTiles**

A boolean that is true if this ImageWriteParam allows tiling grid offset parameters to be set. By default, the value is false. Subclasses must set the value manually.

Subclasses that do not support writing tiles, or that supprt writing but not offsetting tiles must ensure that this value is set to false.

### tileGridXOffset

protected int **tileGridXOffset**

The amount by which the tile grid origin should be offset horizontally from the image origin if tiling has been set, or 0 otherwise. The initial value is 0.

Subclasses that do not support offsetting tiles may ignore this value.

### tileGridYOffset

protected int **tileGridYOffset**

The amount by which the tile grid origin should be offset vertically from the image origin if tiling has been set, or 0 otherwise. The initial value is 0.

Subclasses that do not support offsetting tiles may ignore this value.

### canWriteProgressive

protected boolean **canWriteProgressive**

A boolean that is true if this ImageWriteParam allows images to be written as a progressive sequence of increasing quality passes. By default, the value is false. Subclasses must set the value manually.

Subclasses that do not support progressive encoding must ensure that this value is set to false.

### progressiveMode

protected int **progressiveMode**

The mode controlling progressive encoding, which must be set to one of the four MODE\_\* values, except MODE\_EXPLICIT. The default is MODE\_COPY\_FROM\_METADATA.

Subclasses that do not support progressive encoding may ignore this value.

**See Also:**[MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode())

### canWriteCompressed

protected boolean **canWriteCompressed**

A boolean that is true if this writer can write images using compression. By default, the value is false. Subclasses must set the value manually.

Subclasses that do not support compression must ensure that this value is set to false.

### compressionMode

protected int **compressionMode**

The mode controlling compression settings, which must be set to one of the four MODE\_\* values. The default is MODE\_COPY\_FROM\_METADATA.

Subclasses that do not support compression may ignore this value.

**See Also:**[MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [setCompressionMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int)), [getCompressionMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode())

### compressionTypes

protected [String](http://docs.google.com/java/lang/String.html)[] **compressionTypes**

An array of Strings containing the names of the available compression types. Subclasses must set the value manually.

Subclasses that do not support compression may ignore this value.

### compressionType

protected [String](http://docs.google.com/java/lang/String.html) **compressionType**

A String containing the name of the current compression type, or null if none is set.

Subclasses that do not support compression may ignore this value.

### compressionQuality

protected float **compressionQuality**

A float containing the current compression quality setting. The initial value is 1.0F.

Subclasses that do not support compression may ignore this value.

### locale

protected [Locale](http://docs.google.com/java/util/Locale.html) **locale**

A Locale to be used to localize compression type names and quality descriptions, or null to use a default Locale. Subclasses must set the value manually.

| **Constructor Detail** |
| --- |

### ImageWriteParam

protected **ImageWriteParam**()

Constructs an empty ImageWriteParam. It is up to the subclass to set up the instance variables properly.

### ImageWriteParam

public **ImageWriteParam**([Locale](http://docs.google.com/java/util/Locale.html) locale)

Constructs an ImageWriteParam set to use a given Locale.

**Parameters:**locale - a Locale to be used to localize compression type names and quality descriptions, or null.

| **Method Detail** |
| --- |

### getLocale

public [Locale](http://docs.google.com/java/util/Locale.html) **getLocale**()

Returns the currently set Locale, or null if only a default Locale is supported.

**Returns:**the current Locale, or null.

### canWriteTiles

public boolean **canWriteTiles**()

Returns true if the writer can perform tiling while writing. If this method returns false, then setTiling will throw an UnsupportedOperationException.

**Returns:**true if the writer supports tiling.**See Also:**[canOffsetTiles()](http://docs.google.com/javax/imageio/ImageWriteParam.html#canOffsetTiles()), [setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int))

### canOffsetTiles

public boolean **canOffsetTiles**()

Returns true if the writer can perform tiling with non-zero grid offsets while writing. If this method returns false, then setTiling will throw an UnsupportedOperationException if the grid offset arguments are not both zero. If canWriteTiles returns false, this method will return false as well.

**Returns:**true if the writer supports non-zero tile offsets.**See Also:**[canWriteTiles()](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteTiles()), [setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int))

### setTilingMode

public void **setTilingMode**(int mode)

Determines whether the image will be tiled in the output stream and, if it will, how the tiling parameters will be determined. The modes are interpreted as follows:

* MODE\_DISABLED - The image will not be tiled. setTiling will throw an IllegalStateException.
* MODE\_DEFAULT - The image will be tiled using default parameters. setTiling will throw an IllegalStateException.
* MODE\_EXPLICIT - The image will be tiled according to parameters given in the [setTiling](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)) method. Any previously set tiling parameters are discarded.
* MODE\_COPY\_FROM\_METADATA - The image will conform to the metadata object passed in to a write. setTiling will throw an IllegalStateException.

**Parameters:**mode - The mode to use for tiling. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if canWriteTiles returns false. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if mode is not one of the modes listed above.**See Also:**[setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)), [getTilingMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode())

### getTilingMode

public int **getTilingMode**()

Returns the current tiling mode, if tiling is supported. Otherwise throws an UnsupportedOperationException.

**Returns:**the current tiling mode. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if canWriteTiles returns false.**See Also:**[setTilingMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int))

### getPreferredTileSizes

public [Dimension](http://docs.google.com/java/awt/Dimension.html)[] **getPreferredTileSizes**()

Returns an array of Dimensions indicating the legal size ranges for tiles as they will be encoded in the output file or stream. The returned array is a copy.

The information is returned as a set of pairs; the first element of a pair contains an (inclusive) minimum width and height, and the second element contains an (inclusive) maximum width and height. Together, each pair defines a valid range of sizes. To specify a fixed size, use the same width and height for both elements. To specify an arbitrary range, a value of null is used in place of an actual array of Dimensions.

If no array is specified on the constructor, but tiling is allowed, then this method returns null.

**Returns:**an array of Dimensions with an even length of at least two, or null. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling.

### setTiling

public void **setTiling**(int tileWidth,  
 int tileHeight,  
 int tileGridXOffset,  
 int tileGridYOffset)

Specifies that the image should be tiled in the output stream. The tileWidth and tileHeight parameters specify the width and height of the tiles in the file. If the tile width or height is greater than the width or height of the image, the image is not tiled in that dimension.

If canOffsetTiles returns false, then the tileGridXOffset and tileGridYOffset parameters must be zero.

**Parameters:**tileWidth - the width of each tile.tileHeight - the height of each tile.tileGridXOffset - the horizontal offset of the tile grid.tileGridYOffset - the vertical offset of the tile grid. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling mode is not MODE\_EXPLICIT. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support grid offsets, and the grid offsets are not both zero. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the tile size is not within one of the allowable ranges returned by getPreferredTileSizes. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if tileWidth or tileHeight is less than or equal to 0.**See Also:**[canWriteTiles](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteTiles), [canOffsetTiles](http://docs.google.com/javax/imageio/ImageWriteParam.html#canOffsetTiles), [getTileWidth()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileWidth()), [getTileHeight()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileHeight()), [getTileGridXOffset()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridXOffset()), [getTileGridYOffset()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridYOffset())

### unsetTiling

public void **unsetTiling**()

Removes any previous tile grid parameters specified by calls to setTiling.

The default implementation sets the instance variables tileWidth, tileHeight, tileGridXOffset, and tileGridYOffset to 0.

**Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling mode is not MODE\_EXPLICIT.**See Also:**[setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int))

### getTileWidth

public int **getTileWidth**()

Returns the width of each tile in an image as it will be written to the output stream. If tiling parameters have not been set, an IllegalStateException is thrown.

**Returns:**the tile width to be used for encoding. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling parameters have not been set.**See Also:**[setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)), [getTileHeight()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileHeight())

### getTileHeight

public int **getTileHeight**()

Returns the height of each tile in an image as it will be written to the output stream. If tiling parameters have not been set, an IllegalStateException is thrown.

**Returns:**the tile height to be used for encoding. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling parameters have not been set.**See Also:**[setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)), [getTileWidth()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileWidth())

### getTileGridXOffset

public int **getTileGridXOffset**()

Returns the horizontal tile grid offset of an image as it will be written to the output stream. If tiling parameters have not been set, an IllegalStateException is thrown.

**Returns:**the tile grid X offset to be used for encoding. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling parameters have not been set.**See Also:**[setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)), [getTileGridYOffset()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridYOffset())

### getTileGridYOffset

public int **getTileGridYOffset**()

Returns the vertical tile grid offset of an image as it will be written to the output stream. If tiling parameters have not been set, an IllegalStateException is thrown.

**Returns:**the tile grid Y offset to be used for encoding. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support tiling. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the tiling parameters have not been set.**See Also:**[setTiling(int, int, int, int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)), [getTileGridXOffset()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridXOffset())

### canWriteProgressive

public boolean **canWriteProgressive**()

Returns true if the writer can write out images as a series of passes of progressively increasing quality.

**Returns:**true if the writer supports progressive encoding.**See Also:**[setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode())

### setProgressiveMode

public void **setProgressiveMode**(int mode)

Specifies that the writer is to write the image out in a progressive mode such that the stream will contain a series of scans of increasing quality. If progressive encoding is not supported, an UnsupportedOperationException will be thrown.

The mode argument determines how the progression parameters are chosen, and must be either MODE\_DISABLED, MODE\_COPY\_FROM\_METADATA, or MODE\_DEFAULT. Otherwise an IllegalArgumentException is thrown.

The modes are interpreted as follows:

* MODE\_DISABLED - No progression. Use this to turn off progession.
* MODE\_COPY\_FROM\_METADATA - The output image will use whatever progression parameters are found in the metadata objects passed into the writer.
* MODE\_DEFAULT - The image will be written progressively, with parameters chosen by the writer.

The default is MODE\_COPY\_FROM\_METADATA.

**Parameters:**mode - The mode for setting progression in the output stream. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support progressive encoding. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if mode is not one of the modes listed above.**See Also:**[getProgressiveMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode())

### getProgressiveMode

public int **getProgressiveMode**()

Returns the current mode for writing the stream in a progressive manner.

**Returns:**the current mode for progressive encoding. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support progressive encoding.**See Also:**[setProgressiveMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int))

### canWriteCompressed

public boolean **canWriteCompressed**()

Returns true if this writer supports compression.

**Returns:**true if the writer supports compression.

### setCompressionMode

public void **setCompressionMode**(int mode)

Specifies whether compression is to be performed, and if so how compression parameters are to be determined. The mode argument must be one of the four modes, interpreted as follows:

* MODE\_DISABLED - If the mode is set to MODE\_DISABLED, methods that query or modify the compression type or parameters will throw an IllegalStateException (if compression is normally supported by the plug-in). Some writers, such as JPEG, do not normally offer uncompressed output. In this case, attempting to set the mode to MODE\_DISABLED will throw an UnsupportedOperationException and the mode will not be changed.
* MODE\_EXPLICIT - Compress using the compression type and quality settings specified in this ImageWriteParam. Any previously set compression parameters are discarded.
* MODE\_COPY\_FROM\_METADATA - Use whatever compression parameters are specified in metadata objects passed in to the writer.
* MODE\_DEFAULT - Use default compression parameters.

The default is MODE\_COPY\_FROM\_METADATA.

**Parameters:**mode - The mode for setting compression in the output stream. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression, or does not support the requested mode. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if mode is not one of the modes listed above.**See Also:**[getCompressionMode()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode())

### getCompressionMode

public int **getCompressionMode**()

Returns the current compression mode, if compression is supported.

**Returns:**the current compression mode. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression.**See Also:**[setCompressionMode(int)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int))

### getCompressionTypes

public [String](http://docs.google.com/java/lang/String.html)[] **getCompressionTypes**()

Returns a list of available compression types, as an array or Strings, or null if a compression type may not be chosen using these interfaces. The array returned is a copy.

If the writer only offers a single, mandatory form of compression, it is not necessary to provide any named compression types. Named compression types should only be used where the user is able to make a meaningful choice between different schemes.

The default implementation checks if compression is supported and throws an UnsupportedOperationException if not. Otherwise, it returns a clone of the compressionTypes instance variable if it is non-null, or else returns null.

**Returns:**an array of Strings containing the (non-localized) names of available compression types, or null. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression.

### setCompressionType

public void **setCompressionType**([String](http://docs.google.com/java/lang/String.html) compressionType)

Sets the compression type to one of the values indicated by getCompressionTypes. If a value of null is passed in, any previous setting is removed.

The default implementation checks whether compression is supported and the compression mode is MODE\_EXPLICIT. If so, it calls getCompressionTypes and checks if compressionType is one of the legal values. If it is, the compressionType instance variable is set. If compressionType is null, the instance variable is set without performing any checking.

**Parameters:**compressionType - one of the Strings returned by getCompressionTypes, or null to remove any previous setting. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if there are no settable compression types. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if compressionType is non-null but is not one of the values returned by getCompressionTypes.**See Also:**[getCompressionTypes()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionTypes()), [getCompressionType()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionType()), [unsetCompression()](http://docs.google.com/javax/imageio/ImageWriteParam.html#unsetCompression())

### getCompressionType

public [String](http://docs.google.com/java/lang/String.html) **getCompressionType**()

Returns the currently set compression type, or null if none has been set. The type is returned as a String from among those returned by getCompressionTypes. If no compression type has been set, null is returned.

The default implementation checks whether compression is supported and the compression mode is MODE\_EXPLICIT. If so, it returns the value of the compressionType instance variable.

**Returns:**the current compression type as a String, or null if no type is set. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT.**See Also:**[setCompressionType(java.lang.String)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionType(java.lang.String))

### unsetCompression

public void **unsetCompression**()

Removes any previous compression type and quality settings.

The default implementation sets the instance variable compressionType to null, and the instance variable compressionQuality to 1.0F.

**Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the plug-in does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT.**See Also:**[setCompressionType(java.lang.String)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionType(java.lang.String)), [setCompressionQuality(float)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionQuality(float))

### getLocalizedCompressionTypeName

public [String](http://docs.google.com/java/lang/String.html) **getLocalizedCompressionTypeName**()

Returns a localized version of the name of the current compression type, using the Locale returned by getLocale.

The default implementation checks whether compression is supported and the compression mode is MODE\_EXPLICIT. If so, if compressionType is non-null the value of getCompressionType is returned as a convenience.

**Returns:**a String containing a localized version of the name of the current compression type. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if no compression type is set.

### isCompressionLossless

public boolean **isCompressionLossless**()

Returns true if the current compression type provides lossless compression. If a plug-in provides only one mandatory compression type, then this method may be called without calling setCompressionType first.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks whether compression is supported and the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null true is returned as a convenience.

**Returns:**true if the current compression type is lossless. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the set of legal compression types is non-null and the current compression type is null.

### setCompressionQuality

public void **setCompressionQuality**(float quality)

Sets the compression quality to a value between 0 and 1. Only a single compression quality setting is supported by default; writers can provide extended versions of ImageWriteParam that offer more control. For lossy compression schemes, the compression quality should control the tradeoff between file size and image quality (for example, by choosing quantization tables when writing JPEG images). For lossless schemes, the compression quality may be used to control the tradeoff between file size and time taken to perform the compression (for example, by optimizing row filters and setting the ZLIB compression level when writing PNG images).

A compression quality setting of 0.0 is most generically interpreted as "high compression is important," while a setting of 1.0 is most generically interpreted as "high image quality is important."

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported, and that the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() returns null or compressionType is non-null it sets the compressionQuality instance variable.

**Parameters:**quality - a float between 0and 1 indicating the desired quality level. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the set of legal compression types is non-null and the current compression type is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if quality is not between 0and 1, inclusive.**See Also:**[getCompressionQuality()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQuality())

### getCompressionQuality

public float **getCompressionQuality**()

Returns the current compression quality setting.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported and that the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null, it returns the value of the compressionQuality instance variable.

**Returns:**the current compression quality setting. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the set of legal compression types is non-null and the current compression type is null.**See Also:**[setCompressionQuality(float)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionQuality(float))

### getBitRate

public float **getBitRate**(float quality)

Returns a float indicating an estimate of the number of bits of output data for each bit of input image data at the given quality level. The value will typically lie between 0 and 1, with smaller values indicating more compression. A special value of -1.0F is used to indicate that no estimate is available.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported and the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null, and quality is within bounds, it returns -1.0.

**Parameters:**quality - the quality setting whose bit rate is to be queried. **Returns:**an estimate of the compressed bit rate, or -1.0F if no estimate is available. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the set of legal compression types is non-null and the current compression type is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if quality is not between 0and 1, inclusive.

### getCompressionQualityDescriptions

public [String](http://docs.google.com/java/lang/String.html)[] **getCompressionQualityDescriptions**()

Returns an array of Strings that may be used along with getCompressionQualityValues as part of a user interface for setting or displaying the compression quality level. The String with index i provides a description of the range of quality levels between getCompressionQualityValues[i] and getCompressionQualityValues[i + 1]. Note that the length of the array returned from getCompressionQualityValues will always be one greater than that returned from getCompressionQualityDescriptions.

As an example, the strings "Good", "Better", and "Best" could be associated with the ranges [0, .33), [.33, .66), and [.66, 1.0]. In this case, getCompressionQualityDescriptions would return { "Good", "Better", "Best" } and getCompressionQualityValues would return { 0.0F, .33F, .66F, 1.0F }.

If no descriptions are available, null is returned. If null is returned from getCompressionQualityValues, this method must also return null.

The descriptions should be localized for the Locale returned by getLocale, if it is non-null.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported and that the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null, it returns null.

**Returns:**an array of Strings containing localized descriptions of the compression quality levels. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the set of legal compression types is non-null and the current compression type is null.**See Also:**[getCompressionQualityValues()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityValues())

### getCompressionQualityValues

public float[] **getCompressionQualityValues**()

Returns an array of floats that may be used along with getCompressionQualityDescriptions as part of a user interface for setting or displaying the compression quality level. See [getCompressionQualityDescriptions](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions()) for more information.

If no descriptions are available, null is returned. If null is returned from getCompressionQualityDescriptions, this method must also return null.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported and that the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null, it returns null.

**Returns:**an array of floats indicating the boundaries between the compression quality levels as described by the Strings from getCompressionQualityDescriptions. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the writer does not support compression. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the set of legal compression types is non-null and the current compression type is null.**See Also:**[getCompressionQualityDescriptions()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageWriteParam.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/ImageWriter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/ImageWriteParam.html)    [**NO FRAMES**](http://docs.google.com/ImageWriteParam.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#147n2zr) | [METHOD](#ihv636) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).