| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TransferHandler.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/swing/ToolTipManager.stillInsideTimerAction.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/TransferHandler.DropLocation.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/TransferHandler.html)    [**NO FRAMES**](http://docs.google.com/TransferHandler.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#35nkun2) | [METHOD](#2jxsxqh) |

## **javax.swing**

Class TransferHandler

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.TransferHandler**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html)

public class **TransferHandler**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

This class is used to handle the transfer of a Transferable to and from Swing components. The Transferable is used to represent data that is exchanged via a cut, copy, or paste to/from a clipboard. It is also used in drag-and-drop operations to represent a drag from a component, and a drop to a component. Swing provides functionality that automatically supports cut, copy, and paste keyboard bindings that use the functionality provided by an implementation of this class. Swing also provides functionality that automatically supports drag and drop that uses the functionality provided by an implementation of this class. The Swing developer can concentrate on specifying the semantics of a transfer primarily by setting the transferHandler property on a Swing component.

This class is implemented to provide a default behavior of transferring a component property simply by specifying the name of the property in the constructor. For example, to transfer the foreground color from one component to another either via the clipboard or a drag and drop operation a TransferHandler can be constructed with the string "foreground". The built in support will use the color returned by getForeground as the source of the transfer, and setForeground for the target of a transfer.

Please see  [How to Use Drag and Drop and Data Transfer](http://java.sun.com/docs/books/tutorial/uiswing/misc/dnd.html), a section in *The Java Tutorial*, for more information.

**Since:** 1.4

| **Nested Class Summary** | |
| --- | --- |
| static class | [**TransferHandler.DropLocation**](http://docs.google.com/javax/swing/TransferHandler.DropLocation.html)            Represents a location where dropped data should be inserted. |
| static class | [**TransferHandler.TransferSupport**](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html)            This class encapsulates all relevant details of a clipboard or drag and drop transfer, and also allows for customizing aspects of the drag and drop experience. |

| **Field Summary** | |
| --- | --- |
| static int | [**COPY**](http://docs.google.com/javax/swing/TransferHandler.html#COPY)            An int representing a "copy" transfer action. |
| static int | [**COPY\_OR\_MOVE**](http://docs.google.com/javax/swing/TransferHandler.html#COPY_OR_MOVE)            An int representing a source action capability of either "copy" or "move". |
| static int | [**LINK**](http://docs.google.com/javax/swing/TransferHandler.html#LINK)            An int representing a "link" transfer action. |
| static int | [**MOVE**](http://docs.google.com/javax/swing/TransferHandler.html#MOVE)            An int representing a "move" transfer action. |
| static int | [**NONE**](http://docs.google.com/javax/swing/TransferHandler.html#NONE)            An int representing no transfer action. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**TransferHandler**](http://docs.google.com/javax/swing/TransferHandler.html#TransferHandler())()            Convenience constructor for subclasses. |
|  | [**TransferHandler**](http://docs.google.com/javax/swing/TransferHandler.html#TransferHandler(java.lang.String))([String](http://docs.google.com/java/lang/String.html) property)            Constructs a transfer handler that can transfer a Java Bean property from one component to another via the clipboard or a drag and drop operation. |

| **Method Summary** | |
| --- | --- |
| boolean | [**canImport**](http://docs.google.com/javax/swing/TransferHandler.html#canImport(javax.swing.JComponent,%20java.awt.datatransfer.DataFlavor%5B%5D))([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp, [DataFlavor](http://docs.google.com/java/awt/datatransfer/DataFlavor.html)[] transferFlavors)            Indicates whether a component will accept an import of the given set of data flavors prior to actually attempting to import it. |
| boolean | [**canImport**](http://docs.google.com/javax/swing/TransferHandler.html#canImport(javax.swing.TransferHandler.TransferSupport))([TransferHandler.TransferSupport](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html) support)            This method is called repeatedly during a drag and drop operation to allow the developer to configure properties of, and to return the acceptability of transfers; with a return value of true indicating that the transfer represented by the given TransferSupport (which contains all of the details of the transfer) is acceptable at the current time, and a value of false rejecting the transfer. |
| protected  [Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) | [**createTransferable**](http://docs.google.com/javax/swing/TransferHandler.html#createTransferable(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Creates a Transferable to use as the source for a data transfer. |
| void | [**exportAsDrag**](http://docs.google.com/javax/swing/TransferHandler.html#exportAsDrag(javax.swing.JComponent,%20java.awt.event.InputEvent,%20int))([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp, [InputEvent](http://docs.google.com/java/awt/event/InputEvent.html) e, int action)            Causes the Swing drag support to be initiated. |
| protected  void | [**exportDone**](http://docs.google.com/javax/swing/TransferHandler.html#exportDone(javax.swing.JComponent,%20java.awt.datatransfer.Transferable,%20int))([JComponent](http://docs.google.com/javax/swing/JComponent.html) source, [Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) data, int action)            Invoked after data has been exported. |
| void | [**exportToClipboard**](http://docs.google.com/javax/swing/TransferHandler.html#exportToClipboard(javax.swing.JComponent,%20java.awt.datatransfer.Clipboard,%20int))([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp, [Clipboard](http://docs.google.com/java/awt/datatransfer/Clipboard.html) clip, int action)            Causes a transfer from the given component to the given clipboard. |
| static [Action](http://docs.google.com/javax/swing/Action.html) | [**getCopyAction**](http://docs.google.com/javax/swing/TransferHandler.html#getCopyAction())()            Returns an Action that performs copy operations to the clipboard. |
| static [Action](http://docs.google.com/javax/swing/Action.html) | [**getCutAction**](http://docs.google.com/javax/swing/TransferHandler.html#getCutAction())()            Returns an Action that performs cut operations to the clipboard. |
| static [Action](http://docs.google.com/javax/swing/Action.html) | [**getPasteAction**](http://docs.google.com/javax/swing/TransferHandler.html#getPasteAction())()            Returns an Action that performs paste operations from the clipboard. |
| int | [**getSourceActions**](http://docs.google.com/javax/swing/TransferHandler.html#getSourceActions(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the type of transfer actions supported by the source; any bitwise-OR combination of COPY, MOVE and LINK. |
| [Icon](http://docs.google.com/javax/swing/Icon.html) | [**getVisualRepresentation**](http://docs.google.com/javax/swing/TransferHandler.html#getVisualRepresentation(java.awt.datatransfer.Transferable))([Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) t)            Returns an object that establishes the look of a transfer. |
| boolean | [**importData**](http://docs.google.com/javax/swing/TransferHandler.html#importData(javax.swing.JComponent,%20java.awt.datatransfer.Transferable))([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp, [Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) t)            Causes a transfer to a component from a clipboard or a DND drop operation. |
| boolean | [**importData**](http://docs.google.com/javax/swing/TransferHandler.html#importData(javax.swing.TransferHandler.TransferSupport))([TransferHandler.TransferSupport](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html) support)            Causes a transfer to occur from a clipboard or a drag and drop operation. |

| **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** |
| --- |

### NONE

public static final int **NONE**

An int representing no transfer action.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.TransferHandler.NONE)

### COPY

public static final int **COPY**

An int representing a "copy" transfer action. This value is used when data is copied to a clipboard or copied elsewhere in a drag and drop operation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.TransferHandler.COPY)

### MOVE

public static final int **MOVE**

An int representing a "move" transfer action. This value is used when data is moved to a clipboard (i.e. a cut) or moved elsewhere in a drag and drop operation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.TransferHandler.MOVE)

### COPY\_OR\_MOVE

public static final int **COPY\_OR\_MOVE**

An int representing a source action capability of either "copy" or "move".

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.TransferHandler.COPY_OR_MOVE)

### LINK

public static final int **LINK**

An int representing a "link" transfer action. This value is used to specify that data should be linked in a drag and drop operation.

**Since:** 1.6 **See Also:**[DnDConstants.ACTION\_LINK](http://docs.google.com/java/awt/dnd/DnDConstants.html#ACTION_LINK), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.TransferHandler.LINK)

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

### TransferHandler

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

Constructs a transfer handler that can transfer a Java Bean property from one component to another via the clipboard or a drag and drop operation.

**Parameters:**property - the name of the property to transfer; this can be null if there is no property associated with the transfer handler (a subclass that performs some other kind of transfer, for example)

### TransferHandler

protected **TransferHandler**()

Convenience constructor for subclasses.

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

### getCutAction

public static [Action](http://docs.google.com/javax/swing/Action.html) **getCutAction**()

Returns an Action that performs cut operations to the clipboard. When performed, this action operates on the JComponent source of the ActionEvent by invoking exportToClipboard, with a MOVE action, on the component's TransferHandler.

**Returns:**an Action for performing cuts to the clipboard

### getCopyAction

public static [Action](http://docs.google.com/javax/swing/Action.html) **getCopyAction**()

Returns an Action that performs copy operations to the clipboard. When performed, this action operates on the JComponent source of the ActionEvent by invoking exportToClipboard, with a COPY action, on the component's TransferHandler.

**Returns:**an Action for performing copies to the clipboard

### getPasteAction

public static [Action](http://docs.google.com/javax/swing/Action.html) **getPasteAction**()

Returns an Action that performs paste operations from the clipboard. When performed, this action operates on the JComponent source of the ActionEvent by invoking importData, with the clipboard contents, on the component's TransferHandler.

**Returns:**an Action for performing pastes from the clipboard

### exportAsDrag

public void **exportAsDrag**([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp,  
 [InputEvent](http://docs.google.com/java/awt/event/InputEvent.html) e,  
 int action)

Causes the Swing drag support to be initiated. This is called by the various UI implementations in the javax.swing.plaf.basic package if the dragEnabled property is set on the component. This can be called by custom UI implementations to use the Swing drag support. This method can also be called by a Swing extension written as a subclass of JComponent to take advantage of the Swing drag support.

The transfer *will not necessarily* have been completed at the return of this call (i.e. the call does not block waiting for the drop). The transfer will take place through the Swing implementation of the java.awt.dnd mechanism, requiring no further effort from the developer. The exportDone method will be called when the transfer has completed.

**Parameters:**comp - the component holding the data to be transferred; provided to enable sharing of TransferHandlerse - the event that triggered the transferaction - the transfer action initially requested; either COPY, MOVE or LINK; the DnD system may change the action used during the course of the drag operation

### exportToClipboard

public void **exportToClipboard**([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp,  
 [Clipboard](http://docs.google.com/java/awt/datatransfer/Clipboard.html) clip,  
 int action)  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html)

Causes a transfer from the given component to the given clipboard. This method is called by the default cut and copy actions registered in a component's action map.

The transfer will take place using the java.awt.datatransfer mechanism, requiring no further effort from the developer. Any data transfer *will* be complete and the exportDone method will be called with the action that occurred, before this method returns. Should the clipboard be unavailable when attempting to place data on it, the IllegalStateException thrown by [Clipboard.setContents(Transferable, ClipboardOwner)](http://docs.google.com/java/awt/datatransfer/Clipboard.html#setContents(java.awt.datatransfer.Transferable,%20java.awt.datatransfer.ClipboardOwner)) will be propogated through this method. However, exportDone will first be called with an action of NONE for consistency.

**Parameters:**comp - the component holding the data to be transferred; provided to enable sharing of TransferHandlersclip - the clipboard to transfer the data intoaction - the transfer action requested; this should be a value of either COPY or MOVE; the operation performed is the intersection of the transfer capabilities given by getSourceActions and the requested action; the intersection may result in an action of NONE if the requested action isn't supported **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the clipboard is currently unavailable**See Also:**[Clipboard.setContents(Transferable, ClipboardOwner)](http://docs.google.com/java/awt/datatransfer/Clipboard.html#setContents(java.awt.datatransfer.Transferable,%20java.awt.datatransfer.ClipboardOwner))

### importData

public boolean **importData**([TransferHandler.TransferSupport](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html) support)

Causes a transfer to occur from a clipboard or a drag and drop operation. The Transferable to be imported and the component to transfer to are contained within the TransferSupport.

While the drag and drop implementation calls canImport to determine the suitability of a transfer before calling this method, the implementation of paste does not. As such, it cannot be assumed that the transfer is acceptable upon a call to this method for paste. It is recommended that canImport be explicitly called to cover this case.

Note: The TransferSupport object passed to this method is only valid for the duration of the method call. It is undefined what values it may contain after this method returns.

**Parameters:**support - the object containing the details of the transfer, not null. **Returns:**true if the data was inserted into the component, false otherwise **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if support is null**Since:** 1.6 **See Also:**[canImport(TransferHandler.TransferSupport)](http://docs.google.com/javax/swing/TransferHandler.html#canImport(javax.swing.TransferHandler.TransferSupport))

### importData

public boolean **importData**([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp,  
 [Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) t)

Causes a transfer to a component from a clipboard or a DND drop operation. The Transferable represents the data to be imported into the component.

Note: Swing now calls the newer version of importData that takes a TransferSupport, which in turn calls this method (if the component in the TransferSupport is a JComponent). Developers are encouraged to call and override the newer version as it provides more information (and is the only version that supports use with a TransferHandler set directly on a JFrame or other non-JComponent).

**Parameters:**comp - the component to receive the transfer; provided to enable sharing of TransferHandlerst - the data to import **Returns:**true if the data was inserted into the component, false otherwise**See Also:**[importData(TransferHandler.TransferSupport)](http://docs.google.com/javax/swing/TransferHandler.html#importData(javax.swing.TransferHandler.TransferSupport))

### canImport

public boolean **canImport**([TransferHandler.TransferSupport](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html) support)

This method is called repeatedly during a drag and drop operation to allow the developer to configure properties of, and to return the acceptability of transfers; with a return value of true indicating that the transfer represented by the given TransferSupport (which contains all of the details of the transfer) is acceptable at the current time, and a value of false rejecting the transfer.

For those components that automatically display a drop location during drag and drop, accepting the transfer, by default, tells them to show the drop location. This can be changed by calling setShowDropLocation on the TransferSupport.

By default, when the transfer is accepted, the chosen drop action is that picked by the user via their drag gesture. The developer can override this and choose a different action, from the supported source actions, by calling setDropAction on the TransferSupport.

On every call to canImport, the TransferSupport contains fresh state. As such, any properties set on it must be set on every call. Upon a drop, canImport is called one final time before calling into importData. Any state set on the TransferSupport during that last call will be available in importData.

This method is not called internally in response to paste operations. As such, it is recommended that implementations of importData explicitly call this method for such cases and that this method be prepared to return the suitability of paste operations as well.

Note: The TransferSupport object passed to this method is only valid for the duration of the method call. It is undefined what values it may contain after this method returns.

**Parameters:**support - the object containing the details of the transfer, not null. **Returns:**true if the import can happen, false otherwise **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if support is null**Since:** 1.6 **See Also:**[importData(TransferHandler.TransferSupport)](http://docs.google.com/javax/swing/TransferHandler.html#importData(javax.swing.TransferHandler.TransferSupport)), [TransferHandler.TransferSupport.setShowDropLocation(boolean)](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html#setShowDropLocation(boolean)), [TransferHandler.TransferSupport.setDropAction(int)](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html#setDropAction(int))

### canImport

public boolean **canImport**([JComponent](http://docs.google.com/javax/swing/JComponent.html) comp,  
 [DataFlavor](http://docs.google.com/java/awt/datatransfer/DataFlavor.html)[] transferFlavors)

Indicates whether a component will accept an import of the given set of data flavors prior to actually attempting to import it.

Note: Swing now calls the newer version of canImport that takes a TransferSupport, which in turn calls this method (only if the component in the TransferSupport is a JComponent). Developers are encouraged to call and override the newer version as it provides more information (and is the only version that supports use with a TransferHandler set directly on a JFrame or other non-JComponent).

**Parameters:**comp - the component to receive the transfer; provided to enable sharing of TransferHandlerstransferFlavors - the data formats available **Returns:**true if the data can be inserted into the component, false otherwise**See Also:**[canImport(TransferHandler.TransferSupport)](http://docs.google.com/javax/swing/TransferHandler.html#canImport(javax.swing.TransferHandler.TransferSupport))

### getSourceActions

public int **getSourceActions**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

Returns the type of transfer actions supported by the source; any bitwise-OR combination of COPY, MOVE and LINK.

Some models are not mutable, so a transfer operation of MOVE should not be advertised in that case. Returning NONE disables transfers from the component.

**Parameters:**c - the component holding the data to be transferred; provided to enable sharing of TransferHandlers **Returns:**COPY if the transfer property can be found, otherwise returns NONE

### getVisualRepresentation

public [Icon](http://docs.google.com/javax/swing/Icon.html) **getVisualRepresentation**([Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) t)

Returns an object that establishes the look of a transfer. This is useful for both providing feedback while performing a drag operation and for representing the transfer in a clipboard implementation that has a visual appearance. The implementation of the Icon interface should not alter the graphics clip or alpha level. The icon implementation need not be rectangular or paint all of the bounding rectangle and logic that calls the icons paint method should not assume the all bits are painted. null is a valid return value for this method and indicates there is no visual representation provided. In that case, the calling logic is free to represent the transferable however it wants.

The default Swing logic will not do an alpha blended drag animation if the return is null.

**Parameters:**t - the data to be transferred; this value is expected to have been created by the createTransferable method **Returns:**null, indicating there is no default visual representation

### createTransferable

protected [Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) **createTransferable**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

Creates a Transferable to use as the source for a data transfer. Returns the representation of the data to be transferred, or null if the component's property is null

**Parameters:**c - the component holding the data to be transferred; provided to enable sharing of TransferHandlers **Returns:**the representation of the data to be transferred, or null if the property associated with c is null

### exportDone

protected void **exportDone**([JComponent](http://docs.google.com/javax/swing/JComponent.html) source,  
 [Transferable](http://docs.google.com/java/awt/datatransfer/Transferable.html) data,  
 int action)

Invoked after data has been exported. This method should remove the data that was transferred if the action was MOVE.

This method is implemented to do nothing since MOVE is not a supported action of this implementation (getSourceActions does not include MOVE).

**Parameters:**source - the component that was the source of the datadata - The data that was transferred or possibly null if the action is NONE.action - the actual action that was performed

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TransferHandler.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/swing/ToolTipManager.stillInsideTimerAction.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/TransferHandler.DropLocation.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/TransferHandler.html)    [**NO FRAMES**](http://docs.google.com/TransferHandler.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#35nkun2) | [METHOD](#2jxsxqh) |

[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).