# AWT Desktop Properties

The following refers to standard AWT desktop properties that may be obtained via the  [Toolkit.getDesktopProperty](http://docs.google.com/Toolkit.html#getDesktopProperty(java.lang.String)) method.

Each desktop property is named by a unique string, which is the "name" of that property.

Desktop properties supported by the AWT but not documented elsewhere - typically because there is no suitable method or class - are documented here.

Desktop properties documented elsewhere are those which are tightly coupled with a method or class which documents them.

Since desktop properties abstract an underlying platform setting, they may not be available in environments that do not support them. In the event that a desktop property is unavailable for any reason, the implementation will return null.

The following table summarizes the desktop properties documented here, and their value types.

| Property Name | Value Type | Summary Description |
| --- | --- | --- |
| awt.font.desktophints | [java.util.Map](http://docs.google.com/util/Map.html) | Font smoothing (text antialiasing) settings. |

## Desktop Font Rendering Hints

**Desktop Property:** **"awt.font.desktophints"**

Modern desktops support various forms of text antialiasing (font smoothing).

These are applied by platform-specific heavyweight components. However an application may want to render text using the same text antialiasing on a drawing surface or lightweight (non-platform) component using  [Graphics2D](http://docs.google.com/Graphics2D.html) methods. This is particularly important when creating  [Swing components](http://docs.google.com/javax/swing/JComponent.html) which are required to appear consistent with native desktop components or other Swing components.

### Basic Usage

The standard desktop property named **"awt.font.desktophints"** can be used to obtain the rendering hints that best match the desktop settings. The return value is a  [Map](http://docs.google.com/util/Map.html) of  [RenderingHints](http://docs.google.com/RenderingHints.html) which can be directly applied to a Graphics2D.

It is a Map as more than one hint may be needed. If non-null this can be directly applied to the Graphics2D.

Toolkit tk = Toolkit.getDefaultToolkit();  
Map map = (Map)(tk.getDesktopProperty("awt.font.desktophints"));  
if (map != null) {  
 graphics2D.addRenderingHints(map);  
}

### Advanced Usage Tips

#### Listening for changes

An application can listen for changes in the property using a  [PropertyChangeListener](http://docs.google.com/beans/PropertyChangeListener.html) :

tk.addPropertyChangeListener("awt.font.desktophints", pcl);

Listening for changes is recommended as users can, on rare occasions, reconfigure a desktop environment whilst applications are running in a way that may affect the selection of these hints, and furthermore many desktop environments support dynamic reconfiguration of these running applications to conform to the new settings.

There is no direct way to discover if dynamic reconfiguration is expected of running applications but the default assumption should be that it is expected, since most modern desktop environments do provide this capability.

#### Text Measurement

Text always needs to be measured using the same  [FontRenderContext](http://docs.google.com/font/FontRenderContext.html) as used for rendering. The text anti-aliasing hint is a component of the FontRenderContext. A  [FontMetrics](http://docs.google.com/FontMetrics.html) obtained from the Graphics object on which the hint has been set will measure text appropriately. This is not a unique requirement for clients that specify this hint directly, since the value of the FontRenderContext should never be assumed, so is discussed here principally as a reminder.

#### Saving and restoring Graphics State

Sometimes an application may need to apply these hints on a shared Graphics only temporarily, restoring the previous values after they have been applied to text rendering operations. The following sample code shows one way to do this.

/\*\*  
 \* Get rendering hints from a Graphics instance.  
 \* "hintsToSave" is a Map of RenderingHint key-values.  
 \* For each hint key present in that map, the value of that  
 \* hint is obtained from the Graphics and stored as the value  
 \* for the key in savedHints.  
 \*/  
RenderingHints getRenderingHints(Graphics2D g2d,  
 RenderingHints hintsToSave,  
 RenderingHints savedHints) {  
 if (savedHints == null) {  
 savedHints = new RenderingHints(null);  
 } else {  
 savedHints.clear();  
 }  
 if (hintsToSave.size() == 0) {  
 return savedHints;  
 }  
 /\* RenderingHints.keySet() returns Set