## Specifying Options in Help Resources

Each help resource contains a field in its header that allows you to specify certain options. Notice the options field in the following header component for a menu help resource.

```
resource 'hmnu' (130, "Edit", purgeable) {
    HelpMgrVersion,  // version of Help Manager
    hmDefaultOptions,        // options
    0,                       // balloon definition function
    0,                       // variation code
};
```

You should normally use the hmDefaultOptions constant, as shown in the preceding example, to get the standard behavior for help balloons. However, you can also use the constants that are listed in **Help Manager Data**. (Note that not all options are available for every help resource.)

If you're providing help balloons for a desk accessory or a driver that uses owned resources, use the hmUseSubID constant in the options field. Otherwise, the **Help Manager** treats the resource IDs specified in the rest of your help resource as standard resource IDs. (See the **Resource Manager** for a discussion of owned resources and their resource IDs.)

You often specify tip and rectangle coordinates in your help resources. You might want to use the hmAbsoluteCoords constant when providing help for elements in a scrolling window or whenever the window origin is offset from the origin of the port rectangle. If you specify the hmAbsoluteCoords constant, the **Help Manager** ignores the local coordinates of the port rectangle when tracking the cursor, and instead tracks the mouse position relative to the window origin. When you specify the hmAbsoluteCoords constant as an option in a help resource, the **Help Manager** subtracts the coordinates of the window origin from the coordinates of the mouse position. In addition, the **Help Manager** uses these results for the current mouse position, as shown here:

```
mousepoint.h = mousepoint.h - portRect.left;

mousepoint.v = mousepoint.v - portRect.top;
```

With the hmAbsoluteCoords option specified, the **Help Manager** always assigns coordinates (0,0) to the point in the upper-left corner of the window. So, for example, if the cursor is positioned at point (4,5) in a port rectangle and the window origin is at (3,4), the **Help Manager** calculates the cursor to be at (1,1). If this option is not specified, the **Help Manager** uses the port rectangle's local coordinates when tracking the cursor-for example, when using the **GetMouse** procedure.

The **Help Manager** draws and removes help balloons on screen in three different ways. For all help resources except 'hmnu' resources, the **Help Manager** by default draws and removes help balloons as if they were windows. That is, when drawing a balloon, the **Help Manager**

does not save bits behind the balloon and, when removing the balloon, the **Help Manager** generates an update event. By specifying the hmDefaultOptions constant in your help resources, you always get the standard behavior of help balloons. However, you can often specify two options that change the way balloons are drawn and removed from the screen.

If you specify the hmSaveBitsNoWindow constant in the options field, **Help Manager** does not create a window for displaying the balloon. Instead, the **Help Manager** creates a help balloon that is more like a menu than a window. The **Help Manager** saves the bits behind the balloon when it creates the balloon. When it removes the balloon, the **Help Manager** restores the bits without generating an update event. You should only use this option in a modal environment where the bits behind the balloon cannot change from the time the balloon is drawn to the time it is removed. For example, you might choose the hmSaveBitsNoWindow option in a modal environment when providing help balloons that overlay complex graphics, which might take a long time to redraw with an update event. Note that the **Help Manager** always uses this behavior when drawing and removing help balloons specified in your 'hmnu' resources. That is, when you specify the hmDefaultOptions constant in an 'hmnu' resource, the **Help Manager** provides this sort of balloon instead of drawing a window for a balloon. (In an 'hmnu' resource, you cannot even specify options for drawing a window for a balloon.)

If you specify the hmSaveBitsWindow constant, the **Help Manager** treats the help balloon as a hybrid having properties of both a menu and a window. That is, the **Help Manager** saves the bits behind the balloon when it creates the balloon and, when it removes the balloon, it both restores the bits and generates an update event. You will rarely need this option. It is necessary only in a modal environment that might immediately change to a nonmodal environment-that is, where the bits behind the help balloon are static when the balloon is drawn, but can possibly change before the help balloon is removed. For example, if you use an 'hmnu' resource to provide help balloons for menu titles and menu items, you will notice that the **Help Manager** automatically provides this sort of behavior (even when you do not specify the hmSaveBitsWindow option) when creating help balloons for menu titles.

In the preceding list of constants, the values for the constants represent bit positions that are set to 1. To override more than one default, add the values of the bit positions for the desired options and specify this sum, instead of a constant, in the options field. For example, to use subrange IDs, ignore the window port origin coordinates, and save bits behind the help balloon without generating an update event, you should add the values of the bit positions of these options (1, 2, and 4) and specify their sum (7) in the options field.

If you supply the hmDefaultOptions constant, the **Help Manager** treats the resource IDs in this resource as regular resource IDs and not as subrange IDs; it uses the port rectangle's local coordinates when tracking the cursor; it creates windows when drawing balloons and it generates update events without saving or restoring bits when removing balloons.

The hmMatchInTitle constant is used only in window help ('hwin') resources to match windows containing a specified number of characters in their titles. This constant is explained in more detail in the section called, **Help Balloons in Static Windows** under the heading **Providing Help Balloons for Window Content**.

The following sections describe how to create help resources that provide help balloons for the standard user interface elements of your application.

**Providing Help Balloons for Menus**

**Providing Help Balloons for Items in Dialog and Alert Boxes**

**Providing Help Balloons for Window Content**

**Help Balloons in Static Windows**

**Help Balloons in Dynamic Windows**

**Overriding Help Balloons for Application Icons**

**Overriding Other Default Help Balloons**