

Standard Image-Compression Dialog Components

This chapter discusses standard image-compression dialog components. **Standard image-compression dialog components** provide a consistent user interface for selecting parameters that govern the compression of an image or image sequence and the management of the compression operation. Applications that use these components are freed from many of the details of obtaining and validating image-compression parameters and interacting with the Image Compression Manager to compress an image or sequence.

This chapter is divided into the following sections:

- “About Standard Image-Compression Dialog Components” provides a general introduction to components of this type.
- “Using Standard Image-Compression Dialog Components” discusses the facilities provided to applications by these components.
- “Creating a Standard Image-Compression Dialog Component” describes how to create one of these components.
- “Standard Image-Compression Dialog Components Reference” presents detailed information about the functions that are supported by these components.
- “Summary of Standard Image-Compression Dialog Components” contains a condensed listing of the constants, data structures, and functions supported by these components in C and in Pascal.

If you want to use a standard image-compression dialog component in your application, you should read the first two sections of this chapter, and then use the reference section as appropriate. If you want to create your own standard image-compression dialog component, you should be familiar with all of the information in this chapter.

As components, standard image-compression dialog components rely on the facilities of the Component Manager. In order to use any component, your application must also use the Component Manager. If you are not familiar with this manager, see the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox*. In addition, you should be familiar with image compression in general and the Image Compression Manager in particular. See the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime* for more information.

Note

Throughout this chapter, the term *standard dialog component* refers to the standard image-compression dialog component. The term *standard dialog box* refers to one or both of the two dialog boxes presented by the standard image-compression dialog component. These dialog boxes are shown in Figure 3-1 and Figure 3-2. ♦

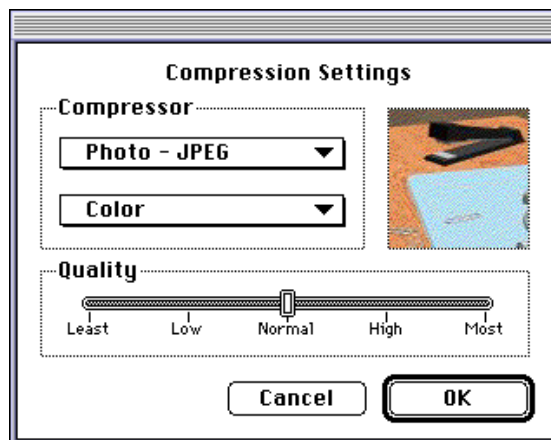
About Standard Image-Compression Dialog Components

Standard image-compression dialog components provide a consistent user interface for specifying the parameters that control the compression of an image or image sequence. Your application specifies a test image for the dialog box and then calls the standard-image compression component. The component then presents a dialog box to the user, manages the dialog box, validates the user's settings, and stores those settings for your application. The standard dialog component also provides numerous facilities for determining reasonable default settings for a given image or sequence. Finally, this component manages the process of compressing the image or image sequence, using the parameter settings provided by the user or your application.

By using a standard image-compression dialog component, you can reduce the amount of work you need to do in your application in order to compress an image or an image sequence. For example, you can eliminate the need to manage interactions with the user and to validate the image-compression parameters specified by the user. Furthermore, the standard dialog component simplifies the process of compressing images or sequences. This, in turn, allows you to focus on the problem at hand, rather than on the details of image-compression parameters. In addition, the standard image-compression dialog component supplied by Apple supports many features that are helpful to the user, including Balloon Help and a test image. Finally, Apple's component will be localized by Apple, so that you need not worry about international issues relating to this dialog box.

Standard image-compression dialog components support two basic dialog boxes. One dialog box provides a minimal interface and is suitable for compressing single images. Figure 3-1 shows an example of this dialog box. Using this dialog box, the user can select a compressor component, the pixel depth for the operation, and the desired spatial quality.

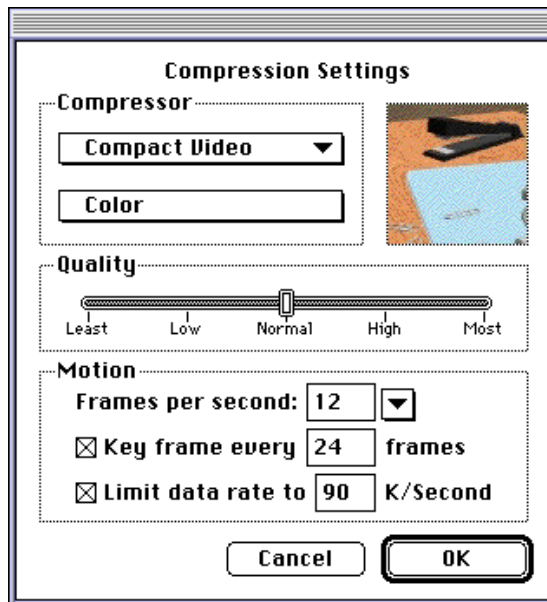
Figure 3-1 Dialog box for single-frame compression



Standard Image-Compression Dialog Components

The other dialog box allows the user to set compression parameters for image sequences. In addition to the parameters supported by the single-frame dialog box, this dialog box supports frame rate, key frame rate, spatial and temporal quality settings, and data rate settings (for more information about these aspects of image compression, see the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime*). Figure 3-2 shows an example of this dialog box.

Figure 3-2 Dialog box for image-sequence compression



Your application can control which dialog box is presented to the user.

By using standard dialog components, you can avoid many of the details of obtaining, validating, and using image-compression parameters. The process of validating image-compression parameters can be very involved, depending upon the capabilities of the selected compressor component. Apple’s standard image-compression dialog component verifies that the user’s settings are valid for the selected compressor. In addition, this component uses a test image to demonstrate the effects of the user’s compression settings.

Using Standard Image-Compression Dialog Components

You can use the standard image-compression dialog component to obtain image or image sequence compression parameters from the user and to manage the process of compressing the image or sequence. This component presents a consistent interface to the user and eliminates the need for you to worry about the details of managing this dialog box. Once you have collected the parameter information from the user, you can use the component to instruct the Image Compression Manager to perform the image or sequence compression. Again, the component manages the details for you.

Because the standard image-compression dialog component is a component, you use the Component Manager to open and close your connection. If you are unfamiliar with components or the Component Manager, see the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox*.

Before you can open a connection to a standard image-compression dialog component, be sure that the Component Manager, Image Compression Manager, and 32-bit Color QuickDraw are available. You can use the Gestalt Manager to determine if these facilities are available. For more information about the Gestalt Manager, see the chapter “Gestalt Manager” in *Inside Macintosh: Operating System Utilities*. For details on 32-bit Color QuickDraw, see the chapter “Color QuickDraw” in *Inside Macintosh: Imaging*.

Once you have established a connection to a standard image-compression dialog component, your application can present the dialog box to the user. The user selects the desired compression parameters and clicks the OK button. The component then stores these parameters for your application, using them, when appropriate, to work with the Image Compression Manager to compress the image or sequence. Figure 3-1 on page 3-4 shows one of the dialog boxes that is supported by the standard image-compression dialog component provided by Apple.

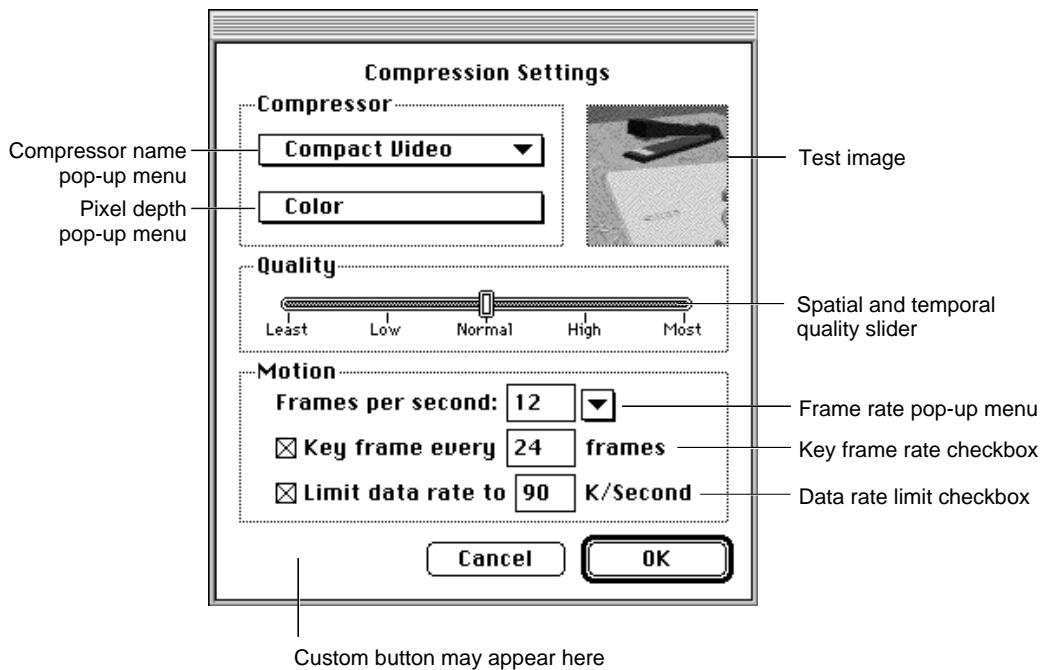
Every standard image-compression dialog box has its own set of parameter information. This information identifies the compressor component to be used, determines which dialog box is used, and specifies the parameters to be used during the compression operation. This information is stored by the component. You can use functions provided by the component to examine or modify these parameters.

The standard image-compression dialog component provided by Apple allows you to augment or extend the interface provided by its dialog boxes. This component supports a single custom button. Your application enables this button when it instructs the component to display the dialog box to the user. You provide the code that supports this

Standard Image-Compression Dialog Components

button in a hook function in your application. In addition, this component allows you to define a filter function—you can use this function to process dialog box events before the component. Figure 3-3 identifies the parts of the dialog box supported by Apple's standard dialog component.

Figure 3-3 Elements of the standard image-compression dialog box



The following sections provide more detailed information about using the standard image-compression dialog component.

- "Opening a Connection to a Standard Image-Compression Dialog Component" tells you how to establish a connection between your application and the standard dialog component.
- "Displaying the Dialog Box to the User" describes the steps you must follow to display the standard dialog box to the user, retrieve the user's settings, and compress an image or sequence.
- "Extending the Basic Dialog Box" discusses several ways your application can customize the basic dialog box.

Opening a Connection to a Standard Image-Compression Dialog Component

As is the case with all components, your application must establish a connection to a standard image-compression dialog component before you can use its services. As with other components, you use the Component Manager's `OpenDefaultComponent` functions to connect to a component. You must use the Component Manager's `CloseComponent` function to close your application's connection when you are done.

Apple provides constants that define the component type and subtype values for standard image-compression dialog components. All of these components have a type value of `'scdi'`; you can use the `StandardCompressionType` constant to specify this value. These components have a subtype value of `'imag'`; the `StandardCompressionSubType` constant defines this value.

Displaying the Dialog Box to the User

Once you have opened a connection to a standard image-compression dialog component, you can proceed to display the dialog box to the user. In preparation, you might establish default parameter settings and specify a test image. Your application may then instruct the component to display the dialog box to the user. The following sections discuss each of these steps in more detail.

Setting Default Parameters

The standard dialog component stores and manages a set of compression parameters for your application. Before presenting the dialog box to the user, you may want to set default values for these parameters. The standard dialog component provides a number of options for establishing these default values:

1. You may supply an image to the component from which it can derive default settings. The component examines the characteristics of the image and sets appropriate default values. The `SCDefaultPictHandleSettings` function works with images stored in picture handles; the `SCDefaultPictFileSettings` function works with images stored in picture files; and the `SCDefaultPixMapSettings` function works with pixel maps. These functions are discussed in “Getting Default Settings for an Image or a Sequence” beginning on page 3-26.
2. If you have not set any defaults, but you do supply a test image for the dialog, the component examines the test image and derives appropriate default values based upon its characteristics. The next section discusses how to assign a test image to the user dialog box.
3. If you have not set any defaults and do not supply a test image, the component uses its own default values.
4. You may modify the settings by using the `SCSetInfo` function, which is described on page 3-36. This function gives you a great deal of freedom—you can use it to modify any of the parameters stored by the component.

If you supply either a test or a default image, the standard dialog component extracts default compression settings from that image, including color table, grayscale information (if appropriate), and compression defaults (if the source image is already compressed). If any of these default values differ from your needs, use the `SCSetInfo` function to modify the value.

Designating a Test Image

The standard image-compression dialog component provided by Apple supports a test image in its dialog box. The component uses this test image to show the user the effect of the current set of compression parameters. Whenever the user changes the dialog box settings, the component applies those parameters to the test image and displays the results in its dialog box. In addition, the standard dialog component may sometimes use the test image to obtain hints about the type of compression operation you expect to perform. In some cases, the component may derive default parameter values by examining the test image.

The component provides three functions that allow you to specify a dialog box's test image. Each of these functions uses a different image source—a handle, a picture file, or a pixel map. Your application is responsible for obtaining the image and for disposing of it after you are done.

The test image portion of the dialog box supported by Apple's standard image-compression dialog component is a square measuring 80 pixels by 80 pixels. In order to deal with test images that are larger than this area, Apple's component allows you to specify a part of the image to display. You can specify an **area of interest**, which indicates a portion of the test image that is to be displayed in the dialog box. If the area of interest is still larger than the display area in the dialog box, the component may shrink the image or crop it (or both) until the image fits.

Listing 3-1 shows one way to specify a test image. This code fragment uses an image that is stored in a picture file. The program asks the user to specify the file, using the `SFGetFilePreview` function. The program then opens the image file and instructs the standard image-compression dialog component to use the picture that is stored in the file.

Listing 3-1 Specifying a test image

```
Point          where;
ComponentInstance ci;
SFTypeList     typeList;
SFReply        inReply;
short          srcPictFRef;

where.h = where.v = -2;          /* center dialog box on the
                                best screen */
typeList[0] = 'PICT';           /* set file type */
```

Standard Image-Compression Dialog Components

```

SFGetFilePreview (where, "\p", nil, 1, typeList, nil,
                  &inReply);
if (!inReply.good) { /* handle error */
}

result = FSOpen (inReply.fName, inReply.vRefNum, &srcPictFRef);
if (result) { /* handle error */
}

result = SCSetTestImagePictFile
        (ci, /* component connection */
         srcPictFRef, /* source picture file */
         nil, /* use the entire image */
         scPreferScalingAndCropping);
/* shrink image and crop it */
if (result) { /* handle error */
}

```

Displaying the Dialog Box and Retrieving Parameters

Standard image-compression dialog components provide two functions that display the dialog box to the user and retrieve the user's compression settings:

`SCRequestImageSettings` and `SCRequestSequenceSettings`. Both of these functions start with your default parameter settings. Any changes made by the user are stored by the component. You may use the `SCGetInfo` function to examine these settings.

The `SCRequestImageSettings` function obtains image-compression parameters from the user and displays the dialog box that is shown in Figure 3-1 on page 3-4. The `SCRequestSequenceSettings` function works with sequence-compression parameters, using the dialog box shown in Figure 3-2 on page 3-5. Both of these functions allow you to augment or extend the interface in the dialog box—see “Extending the Basic Dialog Box,” which begins on page 3-11, for more information about extending the basic dialog boxes.

Listing 3-2 shows how to use the `SCRequestImageSettings` function to display the dialog box to the user and obtain the resulting image-compression settings. This code fragment obtains the compression parameters from the user and then uses those parameters to compress the image that is stored in the file the user selected in Listing 3-1. The program then stores the compressed image in a different file—this fragment assumes that the destination file has already been selected.

Listing 3-2 Displaying the dialog box to the user and compressing an image

```

ComponentInstance    ci;                /* component connection */
short                srcPictFRef;        /* source file */
short                dstPictFRef;        /* destination file */

result = SCRequestImageSettings(ci);
if (result < 0) {                        /* handle error */
}
if (result == scUserCancelled) {         /* user clicked Cancel
                                         button */
}
result = SCCompressPictureFile
        (ci,                            /* component connection */
         srcPictFRef,                    /* source picture file */
         dstPictFRef);                  /* dest picture file */
if (result < 0) {                        /* handle error */
}

```

Note that, because the standard dialog component stores the compression parameters for you, the new user settings become the default values the next time your application interacts with the user. If this is inappropriate, use one of the mechanisms discussed in “Setting Default Parameters” on page 3-8 to modify those defaults.

Extending the Basic Dialog Box

Apple’s standard image-compression dialog component allows you to customize the operation of the user dialog box in a number of ways. First, you can define a filter function. This function, which is a modal-dialog filter function, can process dialog box events before the component does. Your filter function can then perform custom processing that is appropriate to your application. Because the compression dialog box is a movable modal dialog box, you must provide a filter to process update events for your application windows.

Second, you can define a hook function. This function receives item hits before the standard image-compression dialog component does, and can therefore augment the basic dialog box. For example, your hook function can provide additional validation of the user’s selections.

Finally, you can define a custom button in the dialog box. You can then use your hook function to detect when the user clicks this button. Your hook function can then extend the dialog box interface by displaying additional dialog boxes, for example.

Standard Image-Compression Dialog Components

You use the `scExtendedProcsType` request type with the `SCSetInfo` function to take advantage of these mechanisms for customizing the user dialog box. Listing 3-3 contains code that uses this function to define a custom button in the dialog box. Listing 3-4 contains this application's hook function.

Listing 3-3 Defining a custom button in the dialog box

```
SCExtendedProcs ep;

ep.filterProc = MyFilter;      /* custom filter function */
ep.hookProc = MyHook;         /* custom hook function */
ep.refcon = 0;                /* reference constant for filter
                               and hook functions */
BlockMove("\pDefaults", ep.customName, 32);
                               /* custom button name */
SCSetInfo(ci, scExtendedProcsType, &ep);
                               /* set new extended functions */
```

Listing 3-4 shows a hook function that returns the dialog box to its default settings whenever the user clicks the custom button. The standard dialog component calls this function each time the user selects an item in the dialog box. On entry, the hook function receives information about the current dialog box, a pointer to the appropriate standard image-compression dialog parameter block, and a reference constant that is supplied by your application.

This hook function first checks to see whether the user clicked the custom button. If so, the function changes the current compression settings.

Listing 3-4 A sample hook function

```
pascal short MyHook(DialogPtr theDialog, short itemHit,
                    void *params, long refcon)
{
    SCSpatialSettings ss;

    if (itemHit == scCustomItem) { /* check for custom item */
        ss.codecType = 'jpeg';     /* create new settings */
        ss.codec = anyCodec;
        ss.depth = 32;
        ss.spatialQuality = codecNormalQuality;
```

Standard Image-Compression Dialog Components

```

        SCSetInfo(params,          /* component connection */
                  scSpatialSettingsType, /* set spatial settings */
                  &ss);           /* new spatial settings */
    }
    return (itemHit);
}

```

In your hook function, you may want to display additional user dialog boxes.

Apple's standard image-compression dialog component provides two functions that help you position your dialog box on the screen. The `SCPositionDialog` function places a dialog box in a specified location; the `SCPositionRect` function positions a rectangle. By using these functions you can position your dialog boxes near the standard dialog box.

Listing 3-5 contains code that uses the `SCPositionDialog` function to place a Standard File Package dialog box onto the same screen as the standard image-compression dialog box.

Listing 3-5 Positioning related dialog boxes

```

Point      where;          /* positions dialog boxes */
ComponentInstance ci;      /* component connection */

where.h = where.v = -2;    /* center dialog box on the
                           best screen */

result = SCPositionDialog (ci, /* component connection */
                          -3999, /* resource number of dialog box */
                          &where); /* returns upper-left point */

SFPutFile (where,          /* positions the dialog box */
           "\pSave compressed picture as:",
           "\pUntitled",
           nil,
           &outReply);

```

Creating a Standard Image-Compression Dialog Component

Apple's standard image-compression dialog component fully implements the functional interface for components of this type. As a result, this component allows you to customize the dialog box by enabling the custom button or by defining a filter function. In most cases your application should be able to use the component that is supplied by Apple. However, if you want to create your own standard image-compression dialog component, you should read this section.

Apple has defined a component type value for standard image-compression dialog components. All components of this type have the same type and subtype values. You can use the following constants to specify the type and subtype.

```
#define StandardCompressionType      'scdi'
#define StandardCompressionSubType  'imag'
```

Apple has defined a functional interface for standard image-compression dialog components. For information about the functions your component must support, see the next section, "Standard Image-Compression Dialog Components Reference." You can use the following constants to refer to the request codes for each of the functions your component must support.

```
#define scPositionRect          2 /* SCPositionRect */
#define scPositionDialog       3 /* SCPositionDialog */
#define scSetTestImagePictHandle 4 /* SCSetTestImagePictHandle */
#define scSetTestImagePictFile  5 /* SCSetTestImagePictFile */
#define scSetTestImagePixMap    6 /* SCSetTestImagePixMap */
#define scGetBestDeviceRect     7 /* SCGetBestDeviceRect */
#define scRequestImageSettings 10 /* SCRequestImageSettings */
#define scCompressImage         11 /* SCompressImage */
#define scCompressPicture       12 /* SCompressPicture */
#define scCompressPictureFile   13 /* SCompressPictureFile */
#define scRequestSequenceSettings 14 /* SCRequestSequenceSettings */
#define scCompressSequenceBegin 15 /* SCompressSequenceBegin */
#define scCompressSequenceFrame 16 /* SCompressSequenceFrame */
#define scCompressSequenceEnd   17 /* SCompressSequenceEnd */
#define scDefaultPictHandleSettings 18 /* SCDefaultPictHandleSettings */
#define scDefaultPictFileSettings 19 /* SCDefaultPictFileSettings */
#define scDefaultPixMapSettings 20 /* SCDefaultPixMapSettings */
#define scGetInfo               21 /* SCGetInfo */
#define scSetInfo               22 /* SCSetInfo */
#define scNewGWorld             23 /* SCNewGWorld */
```

Standard Image-Compression Dialog Components Reference

This section describes the request types and functions associated with the standard image-compression dialog components and an application-defined function.

Request Types

This section describes the request types used by two standard dialog component functions that allow you to work with the current compression settings for an image or a sequence of images. (You can establish these settings in a number of ways; see “Setting Default Parameters” on page 3-8 for more information about your options.)

You use the `SCGetInfo` function (described on page 3-34) to retrieve settings information. The `SCSetInfo` function (described on page 3-36) enables you to modify the settings.

These functions can work with a number of different types of settings information. When you call either function, you specify the type of data you want to work with. The following request types are defined:

```
#define  scSpatialSettingsType    'sptl'    /* spatial options */
#define  scTemporalSettingsType  'tprl'    /* temporal options */
#define  scDataRateSettingsType  'drat'    /* data rate */
#define  scColorTableType        'clut'    /* color table */
#define  scProgressProcType      'prog'    /* progress function */
#define  scExtendedProcType      'xprc'    /* extended dialog */
#define  scPreferenceFlagsType   'pref'    /* preferences */
#define  scSettingsStateType     'ssta'    /* all settings */
#define  scSequenceIDType        'sequ'    /* sequence ID */
#define  scWindowPositionType    'wndw'    /* window position */
#define  scCodecFlagsType        'cflg'    /* compression flags */
```

Each of these request types requires different parameter data. The following sections discuss each of these request types and their data requirements.

The Spatial Settings Request Type

Use the spatial settings request to retrieve or modify the current spatial compression parameters. These parameters control how each image is compressed.

You supply a pointer to a spatial settings structure. If you are retrieving these settings, the standard dialog component places the current settings into the specified structure; if you are changing the settings, place the new values into the structure—the component uses those values to update its settings.

Standard Image-Compression Dialog Components

The `SCSpatialSettings` data type defines the format and content of the spatial settings structure:

```
typedef struct {
    CodecType      codecType;          /* compressor type */
    CodecComponent codec;              /* compressor */
    short          depth;              /* pixel depth */
    CodecQ         spatialQuality;     /* desired quality */
} SCSpatialSettings;
```

Field descriptions

<code>codecType</code>	<p>Specifies the default compressor type that is displayed in the pop-up menu of compressors in the dialog box. The standard image-compression dialog component uses this field to return the compressor type that was selected by the user.</p> <p>You must set this parameter to one of the compressor types supported by the Image Compression Manager, or to <code>nil</code>.</p> <p>If you set the field to <code>nil</code>, the standard image-compression dialog component uses as the default value the first compressor or compressor type that it retrieves from the Image Compression Manager.</p>
<code>codec</code>	<p>Provides additional information about the default compressor that is displayed in the pop-up menu of compressors in the dialog box. If the user selects a specific compressor component, the standard image-compression dialog component returns the appropriate compressor identifier in this field.</p> <p>The <code>scListEveryCodec</code> bit in the flag in the <code>scPreferenceFlagsType</code> request influences the operation of the compressor list in the dialog box and, therefore, the way the component uses this field.</p> <p>Set the flag to 1 to have the list contain an entry for each compressor component in the system. If the flag is set to 1, the standard image-compression dialog component uses this field along with the <code>codecType</code> field to select the default compressor that appears in the dialog box. To specify a default image compressor component, set this field to the appropriate compressor identifier. When the user clicks OK in the dialog box, the standard image-compression dialog component returns the compressor identifier that corresponds to the selected image compressor component.</p> <p>If you set the field to <code>nil</code>, the standard image-compression dialog component uses as the default value the first compressor of the specified type that it retrieves from the Image Compression Manager.</p>

	<p>If you have set the flag to 0, the list contains only one entry for each type of compressor in the system. The standard image-compression dialog component ignores this field when creating the list of compressor types. In this case, the standard image-compression dialog component does not change the value of this field when the user clicks OK.</p> <p>However, you may use this field to specify additional selection criteria by setting this field to one of the special compressor identifiers supported by the Image Compression Manager (see the chapter “Image Compression Manager” in <i>Inside Macintosh: QuickTime</i> for these special values). The standard image-compression dialog component may use this value when it validates the compression parameters selected by the user.</p>
depth	<p>Specifies the default value of the pixel depth pop-up menu in the dialog box. This menu allows the user to select the color or gray scale resolution value to be used when compressing the image or image sequence. If you set this field to 0, the component chooses an appropriate depth for the default compressor you specified with the <code>theCodec</code> field. See the chapter “Image Compression Manager” in <i>Inside Macintosh: QuickTime</i> for other valid pixel depth values.</p> <p>When the user clicks OK, the standard image-compression dialog component sets this field to the pixel depth value selected by the user. Note that the standard image-compression dialog component may adjust the depth value so that it corresponds to a value that is supported by the compressor that has been selected by the user.</p> <p>The depth returned could be 0 if the <code>scShowBestDepth</code> flag is set.</p>
spatialQuality	<p>Specifies the default setting of the quality slider in the dialog box. This slider controls the spatial quality of the compressed image sequence, which influences the amount of spatial compression that can be achieved. Spatial compression eliminates redundant information within each frame in a sequence. See the chapter “Image Compression Manager” in <i>Inside Macintosh: QuickTime</i> for valid compression quality values.</p> <p>When the user clicks OK, the standard image-compression dialog component sets this field to the spatial quality value selected by the user. Note that the standard image-compression dialog component may adjust the quality value so that it corresponds to a value that is supported by the compressor that has been selected by the user.</p>

The Temporal Settings Request Type

Use the temporal settings request to retrieve or modify the current temporal compression parameters. These parameters govern sequence-compression operations.

You supply a pointer to a temporal settings structure. If you are retrieving these settings, the standard dialog component places the current settings into the specified structure; if you are changing the settings, place the new values into the structure—the component uses those values to update its settings.

Standard Image-Compression Dialog Components

The `SCTemporalSettings` data type defines the format and content of the temporal settings structure:

```
typedef struct {
    CodecQ    temporalQuality;           /* desired quality */
    Fixed     frameRate;                 /* frame rate */
    long      keyFrameRate;              /* key frame rate */
} SCTemporalSettings;
```

Field descriptions`temporalQuality`

Specifies the default setting of the motion quality slider in the dialog box. This slider controls the temporal quality of the compressed image, which influences the amount of temporal compression that can be achieved (note that Apple's component uses the same slider for both spatial and temporal quality). Temporal compression eliminates redundant information between frames in an image sequence. See the chapter "Image Compression Manager" in *Inside Macintosh: QuickTime* for valid compression quality values.

When the user clicks OK, the standard image-compression dialog component sets this field to the temporal quality value selected by the user. Note that the standard image-compression dialog component may adjust the quality value so that it corresponds to a value that is supported by the compressor that has been selected by the user.

`frameRate`

Specifies the default value of the text-edit box that controls the number of frames per second in the image sequence to be compressed. This dialog item allows the user to select the frame rate to be used when compressing the image sequence. Note that this field is stored as a fixed-point number, allowing the user to specify fractional frame rates.

When the user clicks OK, the standard image-compression dialog component sets this field to the frame rate value specified by the user. If you have set the `scAllowZeroFrameRate` flag to 1 in the `scPreferenceFlagsType` request, and the user specifies nothing or 0, the component sets this field to 0.

This dialog item can be useful in cases where your application cannot determine the frame rate of the source movie. For example, movies stored in PICT files do not include frame rate information. Therefore, the user must specify a frame rate for you. Alternatively, some users may want to create movies with different frame rates. This item allows the user to specify a rate for the compressed sequence.

Standard Image-Compression Dialog Components

keyFrameRate Specifies the default value of the text-edit box that controls the frequency with which key frames are inserted into the compressed image sequence. Key frames provide points from which a temporally compressed sequence may be decompressed. For a more complete discussion of key frames, see the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime*.

When the user clicks OK, the standard image-compression dialog component sets this field to the key frame rate value specified by the user. If you have set the `scAllowZeroKeyFrameRate` flag to 1 in the `scPreferenceFlagsType` request, and the user specifies nothing or 0, the component sets this field to 0.

The Data-Rate Settings Request Type

Use the data-rate settings request to retrieve or modify the current temporal compression parameters that govern the data rate. These parameters affect sequence-compression operations.

You supply a pointer to a data-rate settings structure. If you are retrieving these settings, the standard dialog component places the current settings into the specified structure; if you are changing the settings, place the new values into the structure—the component uses those values to update its settings.

The `SCDataRateSettings` data type defines the format and content of the data-rate settings structure:

```
typedef struct {
    long    dataRate;           /* desired data rate */
    long    frameDuration;     /* frame duration */
    CodecQ  minSpatialQuality; /* minimum value */
    CodecQ  minTemporalQuality; /* minimum value */
} SCDataRateSettings;
```

Field descriptions

dataRate Specifies the maximum number of bytes of compressed data your application wants to receive per second. Use this parameter to modulate the rate at which the component passes compressed data to your application. This can be useful to account for hardware limitations during sequence playback.

frameDuration Indicates the duration of each frame, in milliseconds. Set this parameter to 0 to allow the standard dialog component to calculate the duration based upon the frame rate you specify in an `scTemporalSettingsType` request. However, if you allow the user to specify a 0 frame rate (that is, you set the `scAllowZeroFrameRate` flag to 1 in your `scPreferenceFlagsType` request), you must set the frame duration each time you compress a frame, because the component does not have sufficient information to determine an appropriate rate.

Standard Image-Compression Dialog Components

`minSpatialQuality`

Specifies the minimum acceptable spatial quality. In order to meet your specified data rate, the standard dialog component may have to adjust the spatial quality setting. Use this parameter to set a minimum level, which the component may not exceed. See the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime* for values for both this parameter and the `minTemporalQuality` parameter.

`minTemporalQuality`

Specifies the minimum acceptable temporal quality. As with spatial quality, in order to meet your specified data rate, the standard dialog component may have to adjust the temporal quality setting. Use this parameter to set a minimum level, which the component may not exceed.

The Color Table Settings Request Type

Use the color table settings request to retrieve or modify the color table that the standard dialog component uses with all compression operations. Unless you specify otherwise, the component extracts the color table from the source image or sequence.

You supply a pointer to a color table handle (`CTabHandle` data type). Your application is responsible for disposing of this handle when you are done with it. Set the pointer to `nil` to clear the current color table; this may be useful if the current color table is inappropriate for the image or sequence you are working with.

The Progress Function Request Type

Use the progress function request to assign a progress function for use by the standard dialog component. The progress function is a part of your application. The standard dialog component calls this function during time-consuming operations, and reports its progress. Your progress function can use the information it receives from the standard dialog component to keep the user informed about the progress of the operation.

You supply a pointer to an Image Compression Manager progress function structure (see the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime* for information about the format and content of this structure, as well as complete information about progress functions). Set the pointer to `nil` to clear the current progress function; in this case, the standard dialog component does not report its progress to the user. Set the pointer to `-1` to use the component’s default progress function.

The Extended Functions Request Type

Use the extended functions request to extend the interface provided in the standard image or sequence dialog boxes. You may specify a filter function, a hook function, and a custom button; you may retrieve the current settings for these options using the `SCGetInfo` function.

You supply a pointer to an extended functions structure. If you are retrieving these settings, the standard dialog component places the current settings into the specified structure; if you are changing the settings, place the new values into the structure—the component uses those values to update its settings. Set this pointer to `nil` to remove the current functions.

By default, none of these extended interface elements are used.

The `SCEExtendedProcs` data type defines the format and content of the extended functions structure:

```
typedef struct {
    SCModalFilterProcPtr    filterProc; /* filter function */
    SCModalHookProcPtr     hookProc;   /* hook function */
    long                   refcon;     /* reference constant */
    Str31                  customName; /* custom button name */
} SCEExtendedProcs;
```

Field descriptions

filterProc Contains a pointer to a modal-dialog filter function in your application. Because the compression dialog box is a movable modal dialog box, you must provide a filter to process update events for your application windows. The standard component calls your filter function before it processes the event. You can use this function to control events in the dialog box. For example, you might use the filter function to release processing time to other windows displayed by your application while the standard image-compression dialog box is being displayed.

This is how to declare a filter function named `MyFilter`:

```
pascal Boolean MyFilter (DialogPtr theDialog,
                        EventRecord *theEvent, short *itemHit,
                        long refcon);
```

The operation of modal-dialog filter functions is described in the chapter “Dialog Manager” in *Inside Macintosh: Macintosh Toolbox Essentials*. The `refcon` parameter contains the reference constant you supply in the `refcon` field of this structure.

If you do not want to specify a filter function, set this parameter to `nil`.

Standard Image-Compression Dialog Components

hookProc	<p>Contains a pointer to a dialog hook function in your application. The standard component calls your hook function whenever the user selects an item in the dialog box. You can use this function to customize the operation of the standard image-compression dialog box. For example, you might want to support a custom button that activates a secondary dialog box. Another possibility would be to provide additional validation support when the user clicks OK. For an example of defining a custom button, see “Extending the Basic Dialog Box” beginning on page 3-11.</p> <p>This is how to declare a hook function named MyHook:</p> <pre>pascal short MyHook (DialogPtr theDialog, short *itemHit, SCParams *params, long refcon);</pre> <p>The operation of this dialog hook function is described in “Application-Defined Function,” beginning on page 3-45.</p> <p>If you do not want to specify a hook function, set this parameter to nil.</p>
refcon	Specifies a reference constant that is to be passed to the dialog hook function and the modal-dialog filter function.
customName	<p>Specifies the string to be displayed in the custom button in the dialog box.</p> <p>If you are not using a custom button, set this parameter to nil.</p>

The Preference Flags Request Type

Use the preference flags request to specify or retrieve the standard dialog component’s preference flags. These flags govern some of the details of the dialog box that are presented to the user.

You supply a pointer to a long integer. If you are retrieving these flags, the standard dialog component places the current settings into the specified field; if you are changing the flags, set the field with your desired flag values—the component uses those values to update its settings.

By default, the `SCRequestImageSettings` function operates with the `scShowBestDepth` and `scUseMovableModal` flags set to 1. The `SCRequestSequenceSettings` function operates with the `scUseMovableModal` flag set to 1. You should never need to change the values of the `scListEveryCodec` or `scUseMovableModal` flags.

The following flags are defined:

```
#define  scListEveryCodec      (1L<<1)  /* list every component */
#define  scAllowZeroFrameRate (1L<<2)  /* allow 0 frame rate */
#define  scAllowZeroKeyFrameRate
                                     (1L<<3) /* 0 key frame rate OK */
#define  scShowBestDepth      (1L<<4)  /* use best image depth */
#define  scUseMovableModal    (1L<<5)  /* use movable dialog */
```

Flag descriptions

scListEveryCodec

Controls the contents of the pop-up menu of compressors. If you set this flag to 1, the standard image-compression dialog component lists every compressor component that is present in the system. Each entry in the list contains the name of a compressor component. The user may then select a specific component from the list.

If you set this flag to 0, the list contains one entry for each type of compressor component that is present in the system. Each list entry contains the name of a compressor type (for example, a list entry might contain “Animation” for the animation compressor). The user may then select a type of compressor—it is your application’s responsibility to select an appropriate compressor of that type.

scAllowZeroFrameRate

Determines whether the component allows the user to specify a value of 0 for the frame rate. If you set this flag to 1, the component allows the user to specify either 0 or nothing for the frame rate. The component then includes a “best rate” entry in the pop-up menu. If the user specifies 0, the component sets the `frameRate` field in the `SCTemporalSettings` structure to 0. Your application must then determine the best frame rate for the movie.

If you set this flag to 0, the component does not allow the user to enter 0 for the frame rate. In this case, the user must select a specific frame rate.

scAllowZeroKeyFrameRate

Similar to the `scAllowZeroFrameRate` flag, this flag determines whether the component allows the user to specify a value of 0 for the key frame rate. If you set this flag to 1, the component allows the user to specify 0 for the frame rate. If the user specifies 0, the component sets the `keyFrameRate` field in the `SCTemporalSettings` structure to 0. Your application must then determine the best key frame rate for the movie.

If you set this flag to 0, the component does not allow the user to specify 0 for the frame rate. In this case, if the user has enabled temporal compression by checking the key frame checkbox, the user must also select a specific key frame rate.

Standard Image-Compression Dialog Components

`scShowBestDepth`

Determines whether the component includes a “best depth” entry in the pop-up menu for pixel depth. If you set this flag to 1, the component includes a “best depth” entry in the pop-up menu. If the user selects “best depth,” the component sets the depth to 0. Your application must then determine the best pixel depth for the movie. If you set this flag to 0, the component does not include a “best depth” entry in the pop-up menu. The user must select a depth from among the other available choices.

`scUseMovableModal`

Determines whether the standard compression dialog is a movable or a stationary dialog. Set this flag to 1 to create a movable dialog. In this case, you should provide an event filter function to handle update events (use the `scExtendedProcsType` request).

The Settings State Request Type

Use the settings state request to set or retrieve the configuration of the standard dialog component. You may use this request to retrieve the configuration information so that you can save it for later use, or to reconfigure the component based on a saved configuration.

Your application is not concerned with the content of the configuration information that is returned. The standard dialog component saves its configuration in a format that it understands. This request affects only those settings that are valid across system restarts, such as the spatial and temporal compression parameters and the data-rate settings.

You supply a pointer to a handle. When you retrieve the settings, the standard dialog component creates an appropriately-sized handle and places its current configuration information into the handle. Your application is responsible for disposing of the handle when you are done with it.

When you modify the settings, you supply the configuration information in the handle. The component copies the data out of this handle. Your application is responsible for disposing of the handle when you are done with it. Set the pointer to `nil` to reset the component to its default configuration.

The Sequence ID Request Type

Use the sequence ID request type to retrieve the sequence identifier being used by the component’s `SCCompressSequenceFrame` function. You may not use this request to set the sequence identifier.

You supply a pointer to a field of type `ImageSequence` (this is an Image Compression Manager data type). The standard dialog component returns the current sequence identifier in that field.

The Window Position Request Type

Use the window position request to position the user's dialog box.

You supply a pointer to a point. If you are retrieving this information, the standard dialog component places the coordinates of the upper-left corner of the dialog box into this point; if you are changing the dialog box's position, place the new coordinates into the point structure—the component uses those coordinates to position the dialog box.

Normally you should not need to use this request. By default, the standard dialog component centers the dialog box on the screen that is best-suited to display your test image. The component also saves the last window position for movable modal dialogs.

The Control Flags Request Type

Use the control flags request to retrieve or modify the control flags used by the standard dialog component. The standard dialog component passes these flags through to the image compressor it uses to compress your image or sequence. These flags are Image Compression Manager control flags, as described in the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime*.

You supply a pointer to a flags field of data type `CodecFlags` (this is an Image Compression Manager data type). If you are retrieving the flags, the standard dialog component places the current flags into this field. If you are setting new flag values, place your desired settings into the field—the component uses these new flag settings.

By default, the standard dialog component sets all flags to 0 when it compresses still images. When it is compressing sequences, the component sets the `codecFlagsPreviousUpdate` and `codecFlagsUpdatePreviousComp` flags to 1. Typically, you should not need to change these flag settings.

Standard Image-Compression Dialog Component Functions

This section describes the functions that are supported by standard image-compression dialog components. It is divided into the following topics:

- “Getting Default Settings for an Image or a Sequence” discusses how you can use the standard dialog component to derive default compression settings for an image or a sequence.
- “Displaying the Standard Image-Compression Dialog Box” tells you how to present the standard dialog box to the user.
- “Compressing Still Images” discusses functions that allow you to compress still images.
- “Compressing Image Sequences” discusses functions that allow you to compress image sequences.
- “Working With Image or Sequence Settings” describes the functions and data structures you can use to modify the compression settings stored by the standard dialog component.

Standard Image-Compression Dialog Components

- “Specifying a Test Image” tells you how you can specify the image that is displayed to the user in the standard dialog box.
- “Positioning Dialog Boxes and Rectangles” provides information about a number of functions that allow you to position dialog boxes and rectangles that may be related to the standard dialog box.
- “Utility Function” discusses a utility function that the standard dialog component provides to your application.

Getting Default Settings for an Image or a Sequence

This section describes the functions that allow you to derive sensible default compression settings for an image or a sequence. The standard dialog component examines an image you provide and selects appropriate default settings based on the image’s characteristics. The component stores those settings for you and uses them with other functions, including not only functions governing image or sequence compression, but also utility functions such as `SCNewGWorld`. If you choose to display a dialog box to the user, the component uses these settings as the default dialog box settings.

Any of these functions may be used with a single image or an image that is part of a sequence. You tell the standard dialog component whether the image is part of a sequence when you call the function.

If there is a custom color table associated with the image or the sequence, these functions retrieve and store it. You can use the color table settings request (described on page 3-20) to retrieve the custom color table and obtain as much color and depth information as possible from the image or sequence of images.

You can retrieve these settings using the `SCGetInfo` function, or modify them using the `SCSetInfo` function, which are described on page 3-34 and page 3-36, respectively.

There are three functions available: `SCDefaultPictHandleSettings` works with pictures, `SCDefaultPictFileSettings` works with picture files, and `SCDefaultPixMapSettings` works with pixel maps.

SCDefaultPixMapSettings

The `SCDefaultPixMapSettings` function allows you to derive default compression settings for an image that is stored in a pixel map.

```
pascal ComponentResult SCDefaultPixMapSettings
    (ComponentInstance ci, PixMapHandle src,
     short motion);
```

ci Identifies your application’s connection to a standard image-compression dialog component. You obtain this identifier from the Component Manager’s `OpenDefaultComponent` function.

<code>src</code>	Contains a handle to the pixel map to be analyzed.
<code>motion</code>	Specifies whether the image is part of a sequence. Set this parameter to <code>true</code> if the image is part of a sequence; set it to <code>false</code> if you are working with a single still image.

SCDefaultPictHandleSettings

The `SCDefaultPictHandleSettings` function allows you to derive default compression settings for a picture that is stored in a handle.

```
pascal ComponentResult SCDefaultPictHandleSettings
                                (ComponentInstance ci,
                                PicHandle srcPicture,
                                short motion);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from the Component Manager's <code>OpenDefaultComponent</code> function.
<code>srcPicture</code>	Contains a handle to the picture to be analyzed.
<code>motion</code>	Specifies whether the image is part of a sequence. Set this parameter to <code>true</code> if the image is part of a sequence; set it to <code>false</code> if you are working with a single still image.

SCDefaultPictFileSettings

The `SCDefaultPictFileSettings` function allows you to derive default compression settings for a picture that is stored in a file.

```
pascal ComponentResult SCDefaultPictFileSettings
                                (ComponentInstance ci, short srcRef,
                                short motion);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from the Component Manager's <code>OpenDefaultComponent</code> function.
<code>srcRef</code>	Contains a reference to the file to be analyzed.
<code>motion</code>	Specifies whether the image is part of a sequence. Set this parameter to <code>true</code> if the image is part of a sequence; set it to <code>false</code> if you are working with a single still image.

RESULT CODES

File Manager errors

Displaying the Standard Image-Compression Dialog Box

Standard image-compression dialog components provide two functions that allow you to display the standard dialog box to the user and retrieve the compression parameters specified by the user.

Use the `SCRequestImageSettings` function to retrieve the user's preferences for compressing a single image; use the `SCRequestSequenceSettings` functions when you are working with an image sequence.

Both of these functions manipulate the compression settings that the component stores for you. The component may derive the current settings from a number of different sources:

- You may supply an image to the component from which it can derive default settings. You do this by using one of the functions discussed in “Getting Default Settings for an Image or a Sequence” beginning on page 3-26.
- If you have not set any defaults, but you do supply a test image for the dialog, the component examines the test image and derives appropriate default values based upon its characteristics.
- If you have not set any defaults and do not supply a test image, the component uses its own default values.
- You may modify the settings by using the `SCSetInfo` function, which is described on page 3-36.
- You may allow the user to modify those settings by calling one of the functions discussed in this section.

You may customize the dialog boxes by specifying a modal-dialog hook function or a custom button. You may use the custom button to invoke an ancillary dialog box that is specific to your application. See “Request Types” beginning on page 3-15 for more information.

SCRequestImageSettings

The `SCRequestImageSettings` function displays the standard image dialog box to the user; the dialog box is populated with the default settings you have established.

```
pascal ComponentResult SCRequestImageSettings
                                (ComponentInstance ci);
```

`ci` Identifies your application's connection to a standard image-compression dialog component.

DESCRIPTION

The standard dialog component retrieves and validates the user’s selections, and saves the resulting settings for use later.

Use this function when you are working with a single still image.

RESULT CODES

scUserCancelled	1	Dialog box canceled—user clicked Cancel
paramErr	-50	Invalid parameter value

SCRequestSequenceSettings

The SCRequestSequenceSettings function displays the standard sequence dialog box to the user; the dialog box uses the default settings you have established.

```
pascal ComponentResult SCRequestSequenceSettings
                                (ComponentInstance ci);

ci          Identifies your application’s connection to a standard image-compression
              dialog component.
```

DESCRIPTION

The standard dialog component retrieves and validates the user’s selections, and saves the resulting settings for use later.

Use this function when you are working with an image sequence.

RESULT CODES

scUserCancelled	1	Dialog box canceled—user clicked Cancel
paramErr	-50	Invalid parameter value

Compressing Still Images

The standard dialog component provides three functions you may use to compress a still image. These functions differ based on how the image is stored: `SCCompressImage` works with pixel maps; `SCCompressPicture` compresses a picture that is stored in a handle; and `SCCompressPictureFile` works with pictures stored in files.

All of these functions use the current compression settings. See “Displaying the Standard Image-Compression Dialog Box” beginning on page 3-28 for detailed information about establishing these current settings.

If there are no default settings, each of these functions could potentially display the dialog box for single-frame compression operations shown in Figure 3-1 on page 3-4.

SCCompressImage

The `SCCompressImage` function compresses an image that is stored in a pixel map.

```
pascal ComponentResult SCCompressImage (ComponentInstance ci,
                                         PixMapHandle src,
                                         Rect *srcRect,
                                         ImageDescriptionHandle *desc,
                                         Handle *data);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
<code>src</code>	Contains a handle to the pixel map to be compressed.
<code>srcRect</code>	Contains a pointer to a portion of the pixel map to compress. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire pixel map, set this parameter to <code>nil</code> .
<code>desc</code>	Contains a pointer to an image description handle. The standard dialog component creates an image description structure when it compresses the image, and returns a handle to that structure in the field referred to by this parameter. The component sizes that handle appropriately. Your application is responsible for disposing of that handle when you are done with it.
<code>data</code>	Contains a pointer to a handle. The standard dialog component returns a handle to the compressed image data in the field referred to by this parameter. The component sizes that handle appropriately. Your application is responsible for disposing of that handle when you are done with it.

RESULT CODES

`scUserCancelled` 1 Dialog box canceled—user clicked Cancel
 Image Compression Manager errors (from `FCompressImage` function)

SCCompressPicture

The `SCCompressPicture` function compresses a picture that is stored in a handle.

```
pascal ComponentResult SCCompressPicture (ComponentInstance ci,
                                           PicHandle srcPicture,
                                           PicHandle dstPicture);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
-----------------	--

Standard Image-Compression Dialog Components

<code>srcPicture</code>	Contains a handle to the picture to be compressed.
<code>dstPicture</code>	Contains a handle to the compressed picture. The standard dialog component resizes this handle to accommodate the compressed picture. Your application is responsible for creating and disposing of this handle when you are done with it.

RESULT CODES

<code>scUserCancelled</code>	1	Dialog box canceled—user clicked Cancel
Image Compression Manager errors (from <code>FCompressPicture</code> function)		

SCCompressPictureFile

The `SCCompressPictureFile` function compresses a picture that is stored in a file.

```
pascal ComponentResult SCCompressPictureFile
                                (ComponentInstance ci,
                                short srcRefNum, short dstRefNum);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
<code>srcRefNum</code>	Contains a reference to the file to be compressed.
<code>dstRefNum</code>	Contains a reference to the file that is to receive the compressed data. This may be the same as the source file. The standard dialog component places the compressed image data into the file identified by this reference. Your application is responsible for this file after the compression operation.

RESULT CODES

<code>scUserCancelled</code>	1	Dialog box canceled—user clicked Cancel
Image Compression Manager errors (from <code>FCompressPictureFile</code> function)		

Compressing Image Sequences

The standard dialog component provides three functions you may use to compress an image sequence. The `SCCompressSequenceBegin` function allows you to start a sequence-compression operation; use the `SCCompressSequenceFrame` function for each image in the sequence; you end the sequence by calling the `SCCompressSequenceEnd` function. The standard dialog component manages all of the compression details for you. Your application may have only one sequence-compression operation active on any given connection; naturally, you may have more than one connection active at a time.

Standard Image-Compression Dialog Components

All of these functions use the current compression settings. See “Displaying the Standard Image-Compression Dialog Box” beginning on page 3-28 for detailed information about establishing these current settings.

If there are no default settings, each of these functions could potentially display the dialog box for sequence-compression operations shown in Figure 3-2 on page 3-5.

SCCompressSequenceBegin

The `SCCompressSequenceBegin` function initiates a sequence-compression operation. You supply the first image in the sequence so that the component can determine its spatial and graphical characteristics.

```
pascal ComponentResult SCCompressSequenceBegin
                                (ComponentInstance ci,
                                 PixMapHandle src, Rect *srcRect,
                                 ImageDescriptionHandle *desc);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
<code>src</code>	Contains a handle to the pixel map to be compressed. This pixel map must contain the first image in the sequence.
<code>srcRect</code>	Contains a pointer to a portion of the pixel map to compress. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire pixel map, set this parameter to <code>nil</code> .
<code>desc</code>	Contains a pointer to an image description handle. The standard dialog component creates an image description structure when it compresses the image, and returns a handle to that structure in the field referred to by this parameter. The component sizes the handle appropriately. If you do not want this information, set this parameter to <code>nil</code> . The returned structure is valid for the entire sequence. The standard dialog component disposes of the handle when you end the sequence by calling the <code>SCCompressSequenceEnd</code> function. Your application must not dispose of this handle by any other means.

RESULT CODES

Memory Manager errors

Image Compression Manager errors (from `CompressSequenceBegin` function)

SCCompressSequenceFrame

The `SCCompressSequenceFrame` function continues a sequence-compression operation. You must call this function once for each frame in the sequence, including the first frame.

```
pascal ComponentResult SCCompressSequenceFrame
    (ComponentInstance ci, PixMapHandle src,
     Rect *srcRect, Handle *data,
     long *dataSize, short *notSyncFlag);
```

ci	Identifies your application's connection to a standard image-compression dialog component.
src	Contains a handle to the pixel map to be compressed.
srcRect	Contains a pointer to a portion of the pixel map to compress. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire pixel map, set this parameter to <code>nil</code> .
data	Contains a pointer to a handle. The standard dialog component returns a handle to the compressed image data in the field referred to by this parameter. The component sizes that handle appropriately for the sequence. Your application must not dispose of this handle. The standard dialog component disposes of the handle when you end the sequence by calling the <code>SCCompressSequenceEnd</code> function. If you need to lock the handle, be sure to save and restore the handle's state.
dataSize	Contains a pointer to a long integer. The standard dialog component returns a value that indicates the number of bytes of compressed image data that it returns. Note that this value will differ from the size of the handle referred to by the <code>data</code> parameter, because the handle is allocated to accommodate the largest image in the sequence.
notSyncFlag	Contains a pointer to a short integer that indicates whether the compressed frame is a key frame. If the frame is a key frame, the standard dialog component sets the field referred to by this parameter to 0; otherwise, the component sets this field to <code>mediaSampleNotSync</code> . You may use this field to set the <code>sampleFlags</code> parameter of the Movie Toolbox's <code>AddMediaSample</code> function.

RESULT CODES

scUserCancelled	1	Dialog box canceled—user clicked Cancel
-----------------	---	---

Image Compression Manager errors (from `CompressSequenceFrame` function)

SCCompressSequenceEnd

The `SCCompressSequenceEnd` function ends a sequence-compression operation. The standard dialog component disposes of any memory it used to compress the image sequence, including the data and image description buffers. You must call this function once for each sequence you start.

```
pascal ComponentResult SCCompressSequenceEnd
                                (ComponentInstance ci);
```

ci Identifies your application's connection to a standard image-compression dialog component.

Working With Image or Sequence Settings

The standard dialog component provides two functions that allow you to work with the current compression settings for an image or a sequence of images. You can establish these settings in a number of ways: see “Setting Default Parameters” on page 3-8 for more information about your options.

You use the `SCGetInfo` function to retrieve settings information. The `SCSetInfo` function enables you to modify the settings.

These functions can work with a number of different types of settings information. When you call either function, you specify the type of data you want to work with. Each of these request types requires different parameter data. See “Request Types” beginning on page 3-15 for a description of each of these request types and their data requirements.

SCGetInfo

The `SCGetInfo` function allows you to retrieve configuration information from the standard dialog component.

```
pascal ComponentResult SCGetInfo (ComponentInstance ci,
                                OSType type, void *info);
```

ci Identifies your application's connection to a standard image-compression dialog component.

Standard Image-Compression Dialog Components

type	<p>Specifies the type of information you want to retrieve. The following values are valid:</p> <p>scSpatialSettingsType The component returns its spatial compression parameters.</p> <p>scTemporalSettingsType The component returns its temporal compression parameters.</p> <p>scDataRateSettingsType The component returns information about its compression data rate.</p> <p>scColorTableType The component returns its color table.</p> <p>scProgressProcType The component returns a pointer to its progress function.</p> <p>scExtendedProcsType The component returns information about how you have extended the standard dialog box.</p> <p>scPreferenceFlagsType The component returns its current preference flags settings.</p> <p>scSettingsStateType The component returns its complete configuration.</p> <p>scSequenceIDType The component returns its current image-compression sequence identifier.</p> <p>scWindowPositionType The component returns information about where the standard dialog is positioned.</p> <p>scCodecFlagsType The component returns its current image-compression control flags.</p>
info	<p>Contains a pointer to a field that is to receive the information.</p>

DESCRIPTION

You use the `type` parameter to specify the type of information you want to retrieve. The `info` parameter contains a pointer to a location to receive the information (see this section’s introductory text for information about the format of the data that is returned for each request type). If the component cannot satisfy your request, it returns a result code of `scTypeNotFoundErr`.

RESULT CODE

scTypeNotFoundErr	-8971	Component does not have the information you want
-------------------	-------	--

SCSetInfo

The `SCSetInfo` function allows you to modify the standard dialog component's configuration information.

```
pascal ComponentResult SCSetInfo (ComponentInstance ci,
                                   OSType type, void *info);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
<code>type</code>	Specifies the type of information you want to modify. The following values are valid: <ul style="list-style-type: none"> <code>scSpatialSettingsType</code> Modifies the component's spatial compression parameters. <code>scTemporalSettingsType</code> Modifies the component's temporal compression parameters. <code>scDataRateSettingsType</code> Modifies the component's compression data rate. <code>scColorTableType</code> Modifies the component's color table. <code>scProgressProcType</code> Modifies the component's progress function. <code>scExtendedProcsType</code> Allows you to extend the standard dialog box. <code>scPreferenceFlagsType</code> Modifies the component's preference flags settings. <code>scSettingsStateType</code> Configures the component, based on a saved configuration. <code>scWindowPositionType</code> Positions the standard dialog box. <code>scCodecFlagsType</code> Modifies the component's image-compression control flags.
<code>info</code>	Contains a pointer to a field that contains the new configuration information.

DESCRIPTION

You use the `type` parameter to specify the type of information you want to modify. The `info` parameter contains a pointer to a location that contains the new information (see "Request Types" beginning on page 3-15 for information about the format of the data you must supply for each request type). If the component cannot satisfy your request, it returns a result code of `scTypeNotFoundErr`.

RESULT CODE

`scTypeNotFoundErr` -8971 Component does not have the information you want

Specifying a Test Image

The standard image-compression dialog component provided by Apple supports a test image. As you can see in Figure 3-3 on page 3-7, the dialog box contains a small image along with the other parts of the dialog box. The component uses this image to display the effect of the user's image-compression settings. In this manner, the user can experiment with different settings and see the results of those settings immediately.

The component provides three functions that allow you to specify the test image. Use the `SCSetTestImagePictHandle` function if your test image is stored in a handle. Use the `SCSetTestImagePictFile` function if your test image is in a picture file. The `SCSetTestImagePictMap` function sets the test image from a pixel map.

SCSetTestImagePictHandle

The `SCSetTestImagePictHandle` function sets the dialog box's test image from a picture that is stored in a handle.

```
pascal ComponentResult SCSetTestImagePictHandle
    (ComponentInstance ci, PicHandle testPict,
     Rect *testRect, short testFlags);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
<code>testPict</code>	Identifies a handle that contains the new test image. Your application is responsible for disposing of this handle when you are done with it. You must clear the image or close your connection to the standard image-compression dialog component before you dispose of this handle or close the corresponding resource file. You must set this handle as nonpurgeable. Set this parameter to <code>nil</code> to clear the test image.
<code>testRect</code>	Contains a pointer to a rectangle structure. This rectangle specifies, in the coordinate system of the source image, the area of interest or point of interest in the test image. The area of interest defines a portion of the test image that is to be shown to the user in the dialog box. Use this parameter to direct the component to a specific portion of the test image. The component uses the value of the <code>testFlags</code> parameter to determine how it transforms this image before displaying it to the user. The component uses the <code>testFlags</code> parameter only when the test image is larger than the test image portion of the dialog box.

Standard Image-Compression Dialog Components

You may specify a point of interest by setting the points in the rectangle structure so that they enclose a single point—for example, (0,0) and (1,1). The component centers this point in the image that is displayed in the dialog box, and displays the part of the image that fits in the test image portion of the dialog box.

To use the entire picture, specify `nil` in this parameter.

`testFlags` Specifies how the component is to display a test image that is larger than the test image portion of the dialog box. If you set this parameter to 0, the component uses a default method of its own choosing. In all cases, the component centers the area or point of interest in the test image portion of the dialog box, and then displays some part of the test image. You may indicate your display preference by setting this parameter to one of the following values:

`scPreferCropping`

Indicates that the component should crop the test image to fit the test image portion of the dialog box. The component displays the part of the image that fits in the test image portion of the box. If the image is smaller than the space allotted in the dialog box, the component does not alter the image before displaying it—the resulting image is smaller than the available space.

`scPreferScaling`

Indicates that the component should scale the test image to fit the test image portion of the dialog box. The component shrinks the image to fit the test image portion of the dialog box.

`scPreferScalingAndCropping`

Indicates that the component should both scale and crop the test image. This option is useful with very large test images. The component first shrinks the image to approximately the size of the test image portion of the dialog box, and then trims the image so that it fits the available space.

RESULT CODE

`paramErr` -50 Invalid parameter specified

SCSetTestImagePictFile

The `SCSetTestImagePictFile` function sets the dialog box's test image from a picture that is stored in a picture file.

```
pascal ComponentResult SCSetTestImagePictFile
    (ComponentInstance ci, short testFileRef,
     Rect *testRect, short testFlags);
```

ci Identifies your application's connection to a standard image-compression dialog component.

testFileRef Identifies the file that contains the new test image. Your application is responsible for opening this file before calling this function. You must also close the file when you are done with it. You must clear the image or close your connection to the standard image-compression dialog component before you close the file. If the file contains a large image, the component may take some time to display the standard image-compression dialog box. In this case, the component displays the watch cursor while it loads the test image.

Set this parameter to 0 to clear the test image.

testRect Contains a pointer to a rectangle structure. This rectangle specifies, in the coordinate system of the source image, the area of interest or point of interest in the test image. The area of interest defines a portion of the test image that is to be shown to the user in the dialog box. Use this parameter to direct the component to a specific portion of the test image. The component uses the value of the `testFlags` parameter to determine how it transforms large images before displaying them to the user.

You may specify a point of interest by setting the points in the rectangle structure so that they enclose a single point—for example, (0,0) and (1,1). The component centers this point in the image that is displayed in the dialog box, and displays the part of the image that fits in the test image portion of the dialog box.

To use the entire picture file, pass `nil` in this parameter.

testFlags Specifies how the component is to display a test image that is larger than the test image portion of the dialog box. If you set this parameter to 0, the component uses a default method of its own choosing. In all cases, the component centers the area or point of interest in the test image portion of the dialog box, and then displays some part of the test image.

You may indicate your display preference by setting this parameter to one of the following values:

`scPreferCropping`

Indicates that the component should crop the test image to fit the test image portion of the dialog box. The component displays the part of the image that fits in the test image portion of the box. If the image is smaller than the space

Standard Image-Compression Dialog Components

allotted in the dialog box, the component does not alter the image before displaying it—the resulting image is smaller than the available space.

`scPreferScaling`

Indicates that the component should scale the test image to fit the test image portion of the dialog box. The component shrinks the image to fit the test image portion of the dialog box.

`scPreferScalingAndCropping`

Indicates that the component should both scale and crop the test image. This option is useful with very large test images. The component first shrinks the image to approximately the size of the test image portion of the dialog box, then trims the image so that it fits the available space.

RESULT CODES

`paramErr` -50 Invalid parameter specified
File Manager errors

SCSetTestImagePixMap

The `SCSetTestImagePixMap` function sets the dialog box's test image from a picture that is stored in a pixel map.

```
pascal ComponentResult SCSetTestImagePixMap (ComponentInstance ci,
                                             PixMapHandle testPixMap,
                                             Rect *testRect,
                                             short testFlags);
```

`ci` Identifies your application's connection to a standard image-compression dialog component.

`testPixMap` Contains a handle to a pixel map that contains the new test image. Your application is responsible for creating this pixel map before calling this function. You must also dispose of the pixel map when you are done with it. You must clear the image or close your connection to the standard image-compression dialog component before you dispose of the pixel map.

Set this parameter to `nil` to clear the test image.

Standard Image-Compression Dialog Components

<code>testRect</code>	<p>Contains a pointer to a rectangle structure. This rectangle specifies, in the coordinate system of the source image, the area of interest or point of interest in the test image. The area of interest defines a portion of the test image that is to be shown to the user in the dialog box. Use this parameter to direct the component to a specific portion of the test image. The component uses the value of the <code>testFlags</code> parameter to determine how it transforms large images before displaying them to the user.</p> <p>You may specify a point of interest by setting the points in the rectangle structure so that they enclose a single point—for example, (0,0) and (1,1). The component centers this point in the image that is displayed in the dialog box, and displays the part of the image that fits in the test image portion of the dialog box.</p> <p>To use the entire pixel map, specify <code>nil</code> in this parameter.</p>
<code>testFlags</code>	<p>Specifies how the component is to display a test image that is larger than the test image portion of the dialog box. If you set this parameter to 0, the component uses a default method of its own choosing. In all cases, the component centers the area or point of interest in the test image portion of the dialog box, and then displays some part of the test image.</p> <p>You may indicate your display preference by setting this parameter to one of the following values:</p> <p><code>scPreferCropping</code> Indicates that the component should crop the test image to fit the test image portion of the dialog box. The component displays the part of the image that fits in the test image portion of the box. If the image is smaller than the space allotted in the dialog box, the component does not alter the image before displaying it—the resulting image is smaller than the available space.</p> <p><code>scPreferScaling</code> Indicates that the component should scale the test image to fit the test image portion of the dialog box. The component shrinks the image to fit the test image portion of the dialog box.</p> <p><code>scPreferScalingAndCropping</code> Indicates that the component should both scale and crop the test image. This option is useful with very large test images. The component first shrinks the image to approximately the size of the test image portion of the dialog box, then trims the image so that it fits the available space.</p>

RESULT CODE

<code>paramErr</code>	-50	Invalid parameter specified
-----------------------	-----	-----------------------------

Positioning Dialog Boxes and Rectangles

Standard image-compression dialog components provide functions that allow you to position rectangles and dialog boxes. These functions are most useful in helping you to manage dialog boxes that are related to the standard image-compression dialog. For example, your application might support a custom button that initiates a dialog box with the user to specify additional compression parameters. You can use these functions to position that dialog box in relation to the standard image-compression dialog box.

There are two positioning functions: the `SCPositionRect` function positions a rectangle; the `SCPositionDialog` positions a dialog box. The `SCGetBestDeviceRect` function returns information about the best available display device.

SCPositionRect

The `SCPositionRect` function positions a rectangle on the screen. You indicate where you want to put the rectangle by specifying the desired coordinates of the upper-left corner of the rectangle.

```
pascal ComponentResult SCPositionRect (ComponentInstance ci,
                                       Rect *rp, Point *where);
```

ci Identifies your application's connection to a standard image-compression dialog component.

rp Contains a pointer to a rectangle structure. When you call the `SCPositionRect` function, this structure should contain the rectangle's current global coordinates. The `SCPositionRect` function adjusts the coordinates in the structure to reflect the rectangle's new position.

where Contains a pointer to a point in global coordinates identifying the desired location of the upper-left corner of the rectangle. This parameter allows your application to position the rectangle on the screen.

The standard image-compression dialog component supports two special values for this parameter. If you set this parameter to `(-1,-1)`, the component places the rectangle on the display device that has the menu bar. The component centers the rectangle horizontally on that device. The component vertically positions the rectangle so that 1/3 of the vertical space that is not used by the rectangle remains above the rectangle, and the remaining 2/3 of the unused space is below the rectangle.

If you set this parameter to `(-2,-2)`, the component places the rectangle on the display device that supports the highest color or grayscale resolution. The component positions the rectangle as it does for the other special value. This option displays images most clearly and is the recommended value for most cases.

The `SCPositionRect` function adjusts the coordinates of this point to correspond to the upper-left corner of the rectangle.

RESULT CODE

paramErr -50 Invalid parameter specified

SCPositionDialog

The SCSPositionDialog function helps you to position a dialog box on the screen.

```
pascal ComponentResult SCSPositionDialog (ComponentInstance ci,
                                           short id, Point *where);
```

ci	Identifies your application's connection to a standard image-compression dialog component.
id	Specifies the resource number of a 'DLOG' resource. The SCSPositionDialog function positions the dialog box that corresponds to this resource.
where	<p>Contains a pointer to a point in global coordinates identifying the desired location of the upper-left corner of the dialog box. This parameter allows you to indicate how you want to position the dialog box on the screen.</p> <p>The standard image-compression dialog component supports two special values for this parameter. If you set this parameter to (-1,-1), the component places the dialog box on the display device that has the menu bar. The component centers the dialog box horizontally on that device. The component vertically positions the dialog box so that 1/3 of the vertical space that is not used by the box remains above the box, and the remaining 2/3 of the unused space is below the box.</p> <p>If you set this parameter to (-2,-2), the component places the dialog box on the display device that supports the highest color or gray scale resolution. The component positions the dialog box as it does for the other special value. This option displays images most clearly and is the recommended value for most cases.</p> <p>The SCSPositionDialog function adjusts the coordinates of this point to correspond to the upper-left corner of the dialog box.</p>

DESCRIPTION

You indicate where you want to put the dialog box by specifying the desired coordinates of the upper-left corner of the box. The component then derives appropriate location information for the dialog box based upon its size and the display characteristics of the destination device, and returns that location information to your program. You can then pass that information to the Dialog Manager when you want to display the dialog box.

RESULT CODES

paramErr -50 Invalid parameter specified

Resource Manager errors

SCGetBestDeviceRect

The `SCGetBestDeviceRect` function determines the boundary rectangle that surrounds the display device that supports the largest color or grayscale palette.

```
pascal ComponentResult SCGetBestDeviceRect (ComponentInstance ci,
                                             Rect *r);
```

<code>ci</code>	Identifies your application's connection to a standard image-compression dialog component.
<code>r</code>	Contains a pointer to a rectangle structure. The <code>SCGetBestDeviceRect</code> function returns the global coordinates of a rectangle that surrounds the appropriate display device.

DESCRIPTION

The `SCGetBestDeviceRect` function determines the boundary rectangle that surrounds the display device that supports the largest color or grayscale palette. If more than one device supports the same pixel depth, the function returns information about the device that has the highest resolution.

Note that the function subtracts the menu bar from the returned rectangle if the best device is also the main display device.

The standard image-compression dialog component uses this function to position rectangles and dialog boxes when you indicate that the component is to choose the best display device. In general, your application does not need to use this function.

RESULT CODE

<code>paramErr</code>	-50	Invalid parameter specified
-----------------------	-----	-----------------------------

Utility Function

The standard dialog component provides a single utility function that you can use to create a graphics world that is appropriate for the current compression settings. This function is described next.

SCNewGWorld

The SCNewGWorld function creates a graphics world based on the current compression settings.

```
pascal ComponentResult SCNewGWorld (ComponentInstance ci,
                                     GWorldPtr *gwp, Rect *rp,
                                     GWorldFlags flags);
```

ci	Identifies your application's connection to a standard image-compression dialog component.
gwp	Contains a pointer to a pointer to a graphics world. The standard dialog component places a pointer to the new graphics world into the field referred to by this parameter. If the component cannot create the graphics world, it sets this field to nil. Your application is responsible for disposing of the graphics world when you are done with it.
rp	Contains a pointer to the boundaries of the graphics world. If you set this parameter to nil, the standard dialog component uses the test image's boundary rectangle. If you don't specify a boundary rectangle and there is no test image, the component does not create the graphics world.
flags	Contains flags that are passed to QuickDraw's NewGWorld function. See the chapter "Basic QuickDraw" in <i>Inside Macintosh: Imaging</i> for more information about this function.

DESCRIPTION

The SCNewGWorld function creates a graphics world that can accommodate the current compression settings, including color table and grayscale settings (if appropriate). If the selected color table is inappropriate for the pixel depth, the standard dialog component uses a standard color for the depth.

RESULT CODE

scTypeNotFoundErr	-8971	Component cannot create a graphics world
-------------------	-------	--

Application-Defined Function

The standard image-compression dialog component supplied by Apple allows you to extend the interface of the standard dialog box by defining a hook function. This section describes how that hook function operates.

MyHook

This function is called by the standard dialog component whenever the user selects an item in the standard image-compression dialog box. You define the function in your application and assign it to a dialog box with the `hookProc` field of the `scExtendedProcsType` request, which is discussed on page 3-21.

This is how you would define a hook function called `MyHook`:

```
pascal short MyHook (DialogPtr theDialog, short itemHit,
                    void *params, long refcon);
```

<code>theDialog</code>	Contains a pointer to the dialog structure that identifies the current dialog box.
<code>itemHit</code>	Identifies the item clicked by the user.
<code>params</code>	Contains a pointer to a field that contains the identifier for your connection to the standard dialog component. You can use this identifier to call the dialog component's <code>SCGetInfo</code> or <code>SCSetInfo</code> functions.
<code>refcon</code>	Contains the reference constant value you supplied to the <code>SCGetCompressionExtended</code> function.

DESCRIPTION

Your hook function returns a short integer that identifies the item selected by the user. In general, your hook function should return the same item number it receives in the `itemHit` parameter. By returning a specific value, you can affect how the component handles the user selection. The following values are defined:

<code>scOKItem</code>	Indicates that the user clicked the OK button.
<code>scCancelItem</code>	Indicates that the user clicked the Cancel button.
<code>scCustomItem</code>	Indicates that the user clicked the custom button.

If you set the returned value to 0, you cancel the user selection; the dialog box remains on the screen awaiting further action by the user.

The hook function allows your application to tailor or extend the operation of the standard image-compression dialog box. By attaching your hook function to the dialog box, you intercept all user selections. For example, your hook function could perform additional parameter checking whenever the user clicks the OK button. In this case, whenever you detect an incorrect parameter value, you could display a message to the user and then set the returned value to 0, thereby canceling the user's selection. The user would then either cancel the dialog box or try again.

As another example, you could support additional parameters by implementing the dialog box's custom button. You could use your hook function to display a secondary dialog box whenever the user clicks the custom button. For an example of defining and using a custom button, see "Extending the Basic Dialog Box" beginning on page 3-11.

Summary of Standard Image-Compression Dialog Components

C Summary

Constants

```

/* component type value */
#define StandardCompressionType    'scdi' /* standard image-compression
                                           dialog component type */

#define StandardCompressionSubType 'imag' /* standard image-compression
                                           dialog component subtype */

/* preference flags */
#define scListEveryCodec           (1L<<1)    /* list all components */
#define scAllowZeroFrameRate      (1L<<2)    /* allow 0 frame rate */
#define scAllowZeroKeyFrameRate   (1L<<3)    /* allow 0 key frame rate */
#define scShowBestDepth           (1L<<4)    /* allow "best depth" */
#define scUseMovableModal         (1L<<5)    /* use movable dialog */

/* values for testFlags parameter of functions that set test image */
#define scPreferCropping           (1<<0)      /* crop image to fit */
#define scPreferScaling            (1<<1)      /* shrink image to fit */
#define scPreferScalingAndCropping (scPreferScaling + scPreferCropping)
                                           /* shrink then crop */

/* dimensions of the test image portion of the dialog box */
#define scTestImageWidth           80    /* test width of image */
#define scTestImageHeight          80    /* test height of image */

/* possible items returned by hook function */
#define scOKItem                   1    /* user clicked OK */
#define scCancelItem               2    /* user clicked Cancel */
#define scCustomItem               3    /* user clicked custom button */

/* result returned when user canceled */
#define scUserCancelled            1    /* user canceled dialog */

```

Standard Image-Compression Dialog Components

```

/* selectors for standard image-compression dialog components */
#define  scPositionRect          2      /* SCPositionRect */
#define  scPositionDialog        3      /* SCPositionDialog */
#define  scSetTestImagePictHandle 4      /* SCSetTestImagePictHandle */
#define  scSetTestImagePictFile   5      /* SCSetTestImagePictFile */
#define  scSetTestImagePixMap      6      /* SCSetTestImagePixMap */
#define  scGetBestDeviceRect        7      /* SCGetBestDeviceRect */
#define  scRequestImageSettings    10     /* SCRequestImageSettings */
#define  scCompressImage           11     /* SCCompressImage */
#define  scCompressPicture         12     /* SCCompressPicture */
#define  scCompressPictureFile     13     /* SCCompressPictureFile */
#define  scRequestSequenceSettings 14     /* SCRequestSequenceSettings */
#define  scCompressSequenceBegin   15     /* SCCompressSequenceBegin */
#define  scCompressSequenceFrame   16     /* SCCompressSequenceFrame */
#define  scCompressSequenceEnd     17     /* SCCompressSequenceEnd */
#define  scDefaultPictHandleSettings 18    /* SCDefaultPictHandleSettings */
#define  scDefaultPictFileSettings 19     /* SCDefaultPictFileSettings */
#define  scDefaultPixMapSettings   20     /* SCDefaultPixMapSettings */
#define  scGetInfo                 21     /* SCGetInfo */
#define  scSetInfo                 22     /* SCSetInfo */
#define  scNewGWorld               23     /* SCNewGWorld */

/* selectors included for compatibility with earlier linked version
   of standard image-compression dialog component */
#define  scGetCompression          1      /* SCGetCompression */
#define  scShowMotionSettings      (1L<<0) /* SCShowMotionSettings */
#define  scSettingsChangedItem     -1     /* SCSettingsChangedItem */

/* SCSetInfo and SCGetInfo request types */
#define  scSpatialSettingsType     'sptl' /* spatial options */
#define  scTemporalSettingsType    'tprl' /* temporal options */
#define  scDataRateSettingsType    'drat' /* data rate */
#define  scColorTableType          'clut' /* color table */
#define  scProgressProcType        'prog' /* progress function */
#define  scExtendedProcsType       'xprc' /* extended dialog */
#define  scPreferenceFlagsType     'pref' /* preferences */
#define  scSettingsStateType       'ssta' /* all settings */
#define  scSequenceIDType          'sequ' /* sequence ID */
#define  scWindowPositionType      'wndw' /* window position */
#define  scCodecFlagsType          'cflg' /* compression flags */

```

Data Types

```

/* SCModalFilterProcPtr is a pointer to a filter function */
typedef pascal Boolean (*SCModalFilterProcPtr) (DialogPtr theDialog,
        EventRecord *theEvent, short *itemHit, long refcon);

/* SCModalHookProcPtr is a pointer to a hook function */
typedef pascal short (*SCModalHookProcPtr) (DialogPtr theDialog,
        short itemHit, SCParams *params, long refcon);

/* spatial options structure with the spatial settings request */
typedef struct {
    CodecType      codecType;          /* compressor type */
    CodecComponent codec;              /* compressor */
    short          depth;              /* pixel depth */
    CodecQ         spatialQuality;     /* desired quality */
} SCSpatialSettings;

/* temporal options structure with the temporal settings request */
typedef struct {
    CodecQ  temporalQuality;           /* desired quality */
    Fixed   frameRate;                /* frame rate */
    long    keyFrameRate;              /* key frame rate */
} SCTemporalSettings;

/* data rate options with the data rate settings request */
typedef struct {
    long    dataRate;                  /* desired data rate */
    long    frameDuration;             /* frame duration */
    CodecQ  minSpatialQuality;         /* minimum value */
    CodecQ  minTemporalQuality;        /* minimum value */
} SCDataRateSettings;

/* extending the dialog box with the extended functions request */
typedef struct {
    SCModalFilterProcPtr  filterProc; /* filter function */
    SCModalHookProcPtr    hookProc;   /* hook function */
    long                  refcon;      /* reference constant */
    Str31                  customName; /* custom button name */
} SCExtendedProcs;

/* standard compression parameter block for compatibility with earlier
   linked version of standard image-compression dialog components */

```

Standard Image-Compression Dialog Components

```
typedef struct {
    long          flags;           /* control flags */
    CodecType     theCodecType;    /* compressor type */
    CodecComponent theCodec;       /* specific compressor */
    CodecQ        spatialQuality;  /* spatial quality value */
    CodecQ        temporalQuality; /* temporal quality value */
    short         depth;           /* pixel depth */
    Fixed          frameRate;      /* desired frame rate */
    long           keyFrameRate;   /* desired key frame rate */
    long           reserved1;      /* reserved--set to 0) */
    long           reserved2;      /* reserved--set to 0 */
} SCParams;
```

Standard Image-Compression Dialog Component Functions

Getting Default Settings for an Image or a Sequence

```
pascal ComponentResult SCDefaultPixMapSettings
    (ComponentInstance ci, PixMapHandle src,
     short motion);

pascal ComponentResult SCDefaultPictHandleSettings
    (ComponentInstance ci, PicHandle srcPicture,
     short motion);

pascal ComponentResult SCDefaultPictFileSettings
    (ComponentInstance ci, short srcRef,
     short motion);
```

Displaying the Standard Image-Compression Dialog Box

```
pascal ComponentResult SCRequestImageSettings
    (ComponentInstance ci);

pascal ComponentResult SCRequestSequenceSettings
    (ComponentInstance ci);
```

Compressing Still Images

```
pascal ComponentResult SCCompressImage
    (ComponentInstance ci, PixMapHandle src,
     Rect *srcRect, ImageDescriptionHandle *desc,
     Handle *data);

pascal ComponentResult SCCompressPicture
    (ComponentInstance ci, PicHandle srcPicture,
     PicHandle dstPicture);
```


Standard Image-Compression Dialog Components

```
pascal ComponentResult SCCompressPictureFile
    (ComponentInstance ci, short srcRefNum,
     short dstRefNum);
```

Compressing Image Sequences

```
pascal ComponentResult SCCompressSequenceBegin
    (ComponentInstance ci, PixMapHandle src,
     Rect *srcRect, ImageDescriptionHandle *desc);

pascal ComponentResult SCCompressSequenceFrame
    (ComponentInstance ci, PixMapHandle src,
     Rect *srcRect, Handle *data, long *dataSize,
     short *notSyncFlag);

pascal ComponentResult SCCompressSequenceEnd
    (ComponentInstance ci);
```

Working With Image or Sequence Settings

```
pascal ComponentResult SCGetInfo
    (ComponentInstance ci, OSType type, void *info);

pascal ComponentResult SCSetInfo
    (ComponentInstance ci, OSType type, void *info);
```

Specifying a Test Image

```
pascal ComponentResult SCSetTestImagePictHandle
    (ComponentInstance ci, PicHandle testPict,
     Rect *testRect, short testFlags);

pascal ComponentResult SCSetTestImagePictFile
    (ComponentInstance ci, short testFileRef,
     Rect *testRect, short testFlags);

pascal ComponentResult SCSetTestImagePixMap
    (ComponentInstance ci, PixMapHandle testPixMap,
     Rect *testRect, short testFlags);
```

Positioning Dialog Boxes and Rectangles

```
pascal ComponentResult SCPositionRect
    (ComponentInstance ci, Rect *rp, Point *where);

pascal ComponentResult SCPositionDialog
    (ComponentInstance ci, short id, Point *where);

pascal ComponentResult SCGetBestDeviceRect
    (ComponentInstance ci, Rect *r);
```

Standard Image-Compression Dialog Components

Utility Function

```
pascal ComponentResult SCNewGWorld
                                (ComponentInstance ci, GWorldPtr *gwp,
                                 Rect *rp, GWorldFlags flags);
```

Application-Defined Function

```
pascal short MyHook              (DialogPtr theDialog, short itemHit,
                                  void *params, long refcon);
```

Pascal Summary

Constants

```
CONST
    {component type value}
    StandardCompressionType      = 'scdi';  {standard image-compression }
                                         { dialog component type}

    StandardCompressionSubType = 'imag';  {standard image-compression }
                                         { dialog component subtype}

    {preference flags}
    scListEveryCodec              = $2;      {list all components}
    scAllowZeroFrameRate          = $4;      {allow 0 frame rate}
    scAllowZeroKeyFrameRate       = $8;      {allow 0 key frame rate}
    scShowBestDepth               = $10;     {allow "best depth"}
    scUseMovableModal             = $20;     {use movable dialog box}

    {values for testFlags parameter of functions that set test image}
    scPreferCropping              = 1;      {crop image to fit}
    scPreferScaling               = 2;      {shrink image to fit}
    scPreferScalingAndCropping    = 3;      {shrink then crop}

    {dimensions of the test image portion of the dialog box}
    scTestImageWidth              = 80;     {test width of image}
    scTestImageHeight             = 80;     {test height of image}

    {possible items returned by hook function}
    scOKItem                      = 1;      {user clicked OK}
    scCancelItem                  = 2;      {user clicked Cancel}
    scCustomItem                  = 3;      {user clicked custom button}
```

Standard Image-Compression Dialog Components

```

{result returned when user canceled}
scUserCancelled    = 1; {user canceled dialog}

{selectors for standard image-compression dialog components}
kScPositionRect          = 2;    {SCPositionRect}
kScPositionDialog        = 3;    {SCPositionDialog}
kScSetTestImagePictHandle = 4;    {SCSetTestImagePictHandle}
kScSetTestImagePictFile   = 5;    {SCSetTestImagePictFile}
kScSetTestImagePixMap     = 6;    {SCSetTestImagePixMap}
kScGetBestDeviceRect      = 7;    {SCGetBestDeviceRect}
kScRequestImageSettings   = $A;   {SCRequestImageSettings}
kScCompressImage          = $B;   {SCCompressImage}
kScCompressPicture        = $C;   {SCCompressPicture}
kScCompressPictureFile    = $D;   {SCCompressPictureFile}
kScRequestSequenceSettings = $E;   {SCRequestSequenceSettings}
kScCompressSequenceBegin  = $F;   {SCCompressSequenceBegin}
kScCompressSequenceFrame  = $10;  {SCCompressSequenceFrame}
kScCompressSequenceEnd    = $11;  {SCCompressSequenceEnd}
kScDefaultPictHandleSettings = $12; {SCDefaultPictHandleSettings}
kScDefaultPictFileSettings = $13;  {SCDefaultPictFileSettings}
kScDefaultPixMapSettings  = $14;  {SCDefaultPixMapSettings}
kScGetInfo                = $15;  {SCGetInfo}
kScSetInfo                = $16;  {SCSetInfo}
kScNewGWorld              = $17;  {SCNewGWorld}

{selectors included for compatibility with earlier linked version }
{ of standard image-compression dialog component}
kScShowMotionSettings     = 1; {SCShowMotionSettings}
kScGetCompression         = 1; {SCGetCompression}
kScSettingsChangedItem    = -1; {SCSettingsChangedItem}

{SCSetInfo and SCGetInfo request types}
scSpatialSettingsType     = 'sptl'; {spatial options}
scTemporalSettingsType    = 'tprl'; {temporal options}
scDataRateSettingsType    = 'drat'; {data rate}
scColorTableType          = 'clut'; {color table}
scProgressProcType        = 'prog'; {progress function}
scExtendedProcsType       = 'xprc'; {extended dialog}
scPreferenceFlagsType     = 'pref'; {preferences}
scSettingsStateType       = 'sstl'; {all settings}
scSequenceIDType          = 'sequ'; {sequence ID}
scWindowPositionType      = 'wndw'; {window position}
scCodecFlagsType          = 'cflg'; {compression flags}

```

Data Types

TYPE

```
{SCModalFilterProcPtr is a pointer to a filter function}
SCModalFilterProcPtr = ProcPtr;

{SCModalHookProcPtr is a pointer to a hook function}
SCModalHookProcPtr = ProcPtr;

{spatial options structure with the spatial settings request}
SCSpatialSettings =
RECORD
    cType:           CodecType;           {compressor type}
    codec:           CodecComponent;       {compressor}
    depth:           Integer;              {pixel depth}
    spatialQuality:   CodecQ;              {desired quality}
END;

{temporal options structure with the temporal settings request}
SCTemporalSettings =
RECORD
    temporalQuality: CodecQ;              {desired quality}
    frameRate:       Fixed;               {frame rate}
    keyFrameRate:    LongInt;             {key frame rate}
END;

{data rate options with the data rate settings request}
SCDataRateSettings =
RECORD
    dataRate:        LongInt;             {desired data rate}
    frameDuration:    LongInt;             {frame duration}
    minSpatialQuality: CodecQ;             {minimum value}
    minTemporalQuality: CodecQ;           {minimum value}
END;

{extending the dialog box with the extended functions request}
SCExtendedProcs =
RECORD
    filterProc:      SCModalFilterProcPtr; {filter function}
    hookProc:        SCModalHookProcPtr;   {hook function}
    refCon:          LongInt;               {reference constant}
    customName:      Str31;                {custom button name}
END;
```

Standard Image-Compression Dialog Components

```

{standard compression parameter block included for compatibility }
{ with earlier linked version of standard-image compression dialog }
{ component}
SCParams =
RECORD
    flags :           LongInt;           {control flags}
    theCodecType:      CodecType;         {compressor type}
    theCodec:          CodecComponent;    {specific compressor}
    spatialQuality:    CodecQ;            {spatial quality value}
    temporalQuality:   CodecQ;            {temporal quality value}
    depth:             Integer;           {pixel depth}
    frameRate:         Fixed;             {desired frame rate}
    keyFrameRate:      LongInt;           {desired key frame rate}
    reserved1:         LongInt;           {reserved--set to 0}
    reserved2:         LongInt;           [reserved--set to 0]
END;
```

Standard Image-Compression Dialog Component Routines

Getting Default Settings for an Image or a Sequence

```

FUNCTION SCDefaultPixMapSettings
    (ci: ComponentInstance; src: PixMapHandle;
     motion: Boolean): ComponentResult;

FUNCTION SCDefaultPictHandleSettings
    (ci: ComponentInstance; src: PicHandle;
     motion: Boolean): ComponentResult;

FUNCTION SCDefaultPictFileSettings
    (ci: ComponentInstance; srcRef: Integer;
     motion: Boolean): ComponentResult;
```

Displaying the Standard Image-Compression Dialog Box

```

FUNCTION SCRequestImageSettings
    (ci: ComponentInstance): ComponentResult;

FUNCTION SCRequestSequenceSettings
    (ci: ComponentInstance): ComponentResult;
```

Compressing Still Images

```

FUNCTION SCCompressImage    (ci: ComponentInstance; src: PixMapHandle;
                             srcRect: Rect;
                             VAR desc: ImageDescriptionHandle;
                             VAR data: Handle): ComponentResult;
```

Standard Image-Compression Dialog Components

```

FUNCTION SCCompressPicture (ci: ComponentInstance; src, dst: PicHandle):
    ComponentResult;

FUNCTION SCCompressPictureFile
    (ci: ComponentInstance; srcRef,
     dstRef: Integer): ComponentResult;

```

Compressing Image Sequences

```

FUNCTION SCCompressSequenceBegin
    (ci: ComponentInstance; src: PixMapHandle;
     srcRect: Rect;
     VAR desc: ImageDescriptionHandle):
    ComponentResult;

FUNCTION SCCompressSequenceFrame
    (ci: ComponentInstance; src: PixMapHandle;
     srcRect: Rect; VAR data: Handle;
     VAR dataSize: LongInt;
     VAR notSyncFlag: Boolean): ComponentResult;

FUNCTION SCCompressSequenceEnd
    (ci: ComponentInstance): ComponentResult;

```

Working With Image or Sequence Settings

```

FUNCTION SCGetInfo (ci: ComponentInstance; infoType: OSType;
    info: Ptr): ComponentResult;

FUNCTION SCSetInfo (ci: ComponentInstance; infoType: OSType;
    info: Ptr): ComponentResult;

```

Specifying a Test Image

```

FUNCTION SCSetTestImagePictHandle
    (ci: ComponentInstance; testPict: PicHandle;
     testRect: RectPtr; testFlags: Integer):
    ComponentResult;

FUNCTION SCSetTestImagePictFile
    (ci: ComponentInstance; testFileRef: Integer;
     testRect: RectPtr; testFlags: Integer):
    ComponentResult;

FUNCTION SCSetTestImagePixMap
    (ci: ComponentInstance;
     testPixMap: PixMapHandle; testRect: RectPtr;
     testFlags: Integer): ComponentResult;

```

Positioning Dialog Boxes and Rectangles

```
FUNCTION SCPositionRect      (ci: ComponentInstance; r: RectPtr;  
                             VAR where: Point): ComponentResult;  
  
FUNCTION SCPositionDialog   (ci: ComponentInstance; id: Integer;  
                             VAR where: Point): ComponentResult;  
  
FUNCTION SCGetBestDeviceRect  
                             (ci: ComponentInstance; r: RectPtr):  
                             ComponentResult;
```

Utility Function

```
FUNCTION SCNewGWorld        (ci: ComponentInstance; VAR gwp: GWorldPtr;  
                             VAR rp: Rect; flags: GWorldFlags):  
                             ComponentResult;
```

Application-Defined Routine

```
FUNCTION MyHook              (theDialog: DialogPtr; itemHit: Integer;  
                             params Ptr; refcon: LongInt): Integer;
```

Result Codes

scTypeNotFoundErr	-8971	Component does not have the information you want
-------------------	-------	--

