CIFilter Class Reference

Cocoa > Graphics & Imaging

2007-12-11

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

CIFilter Class Reference 5

Overview 5
Tasks 5
- Creating a Filter 5
- Creating a Filter from a RAW Image 6
- Accessing Registered Filters 6
- Registering a Filter 6
- Getting Filter Parameters and Attributes 6
- Setting Default Values 6
- Applying a Filter 6
- Getting Localized Information for Registered Filters 7

Class Methods 7
- filterNamesInCategories: 7
- filterNamesInCategory: 8
- filterWithImageData:options: 8
- filterWithImageURL:options: 9
- filterWithName: 9
- filterWithName:keysAndValues: 10
- localizedDescriptionForFilterName: 10
- localizedNameForCategory: 11
- localizedNameForFilterName: 11
- localizedReferenceDocumentationForFilterName: 12
- registerFilterName:constructor:classAttributes: 12

Instance Methods 13
- apply: 13
- apply:arguments:options: 13
- attributes 14
- inputKeys 15
- outputKeys 16
- setDefaults 16

Constants 16
- Filter Attribute Keys 16
- Data Type Attributes 19
- Vector Quantity Attributes 20
- Color Attribute Keys 20
- Filter Category Keys 21
- Options for Applying a Filter 24
- User Interface Control Options 25
- Filter Parameter Keys 25
- RAW Image Options 29

Document Revision History 33

Index 35

CIFilter Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Availability	Mac OS X v10.4 and later
Declared in	CIFilter.h CIRAWFilter.h
Companion guides	Core Image Programming Guide Image Unit Tutorial Core Image Filter Reference
Related sample code	CarbonCocoaCoreImageTab CIAnnotation CIVideoDemoGL QTCoreImage101 Reducer

Overview

The `CIFilter` class produces a `CIImage` object as output. Typically, a filter takes one or more images as input. Some filters, however, generate an image based on other types of input parameters. The parameters of a `CIFilter` object are set and retrieved through the use of key-value pairs.

You use the `CIFilter` object in conjunction with other Core Image classes, such as `CIImage`, `CIContext`, `CIImageAccumulator`, and `CIColor`, to take advantage of the built-in Core Image filters when processing images, creating filter generators, or writing custom filters.

Tasks

Creating a Filter

`+ filterWithName:` (page 9)
Creates a `CIFilter` object for a specific kind of filter.

`+ filterWithName:keysAndValues:` (page 10)
Creates a `CIFilter` object for a specific kind of filter and initializes the input values.

Creating a Filter from a RAW Image

`+ filterWithImageData:options:` (page 8)
Returns a `CIFilter` object initialized with RAW image data supplied to the method.

`+ filterWithImageURL:options:` (page 9)
Returns a `CIFilter` object initialized with data from a RAW image file.

Accessing Registered Filters

`+ filterNamesInCategories:` (page 7)
Returns an array of all published filter names that match all the specified categories.

`+ filterNamesInCategory:` (page 8)
Returns an array of all published filter names in the specified category.

Registering a Filter

`+ registerFilterName:constructor:classAttributes:` (page 12)
Publishes a custom filter that is not packaged as an image unit.

Getting Filter Parameters and Attributes

`- attributes` (page 14)
Returns a dictionary of key-value pairs that describe the filter.

`- inputKeys` (page 15)
Returns an array that contains the names of the input parameters to the filter.

`- outputKeys` (page 16)
Returns an array that contains the names of the output parameters for the filter.

Setting Default Values

`- setDefaults` (page 16)
Sets all input values for a filter to default values.

Applying a Filter

`- apply:arguments:options:` (page 13)
Produces a *`CIImage`* object by applying arguments to a kernel function and using options to control how the kernel function is evaluated.

`- apply:` (page 13)
Produces a `CIImage` object by applying a kernel function.

Getting Localized Information for Registered Filters

`+ localizedNameForFilterName:` (page 11)
Returns the localized name for the specified filter name.

`+ localizedNameForCategory:` (page 11)
Returns the localized name for the specified filter category.

`+ localizedDescriptionForFilterName:` (page 10)
Returns the localized description of a filter for display in the user interface.

`+ localizedReferenceDocumentationForFilterName:` (page 12)
Returns the location of the localized reference documentation that describes the filter.

Class Methods

filterNamesInCategories:

Returns an array of all published filter names that match all the specified categories.

```
+ (NSArray *)filterNamesInCategories:(NSArray *)categories
```

Parameters

categories
One or more filter categories. Pass `nil` to get all filters in all categories.

Return Value
An array that contains all published filter names that match all the categories specified by the `categories` argument.

Discussion
When you pass more than one filter category, this method returns the intersection of the filters in the categories. For example, if you pass the categories `kCICategoryBuiltIn` (page 24) and `kCICategoryFilterGenerator` (page 24), you obtain all the filters that are members of both the built-in and generator categories. But if you pass in `kCICategoryGenerator` and `kCICategoryStylize` (page 23), you will not get any filters returned to you because there are no filters that are members of both the generator and stylize categories. If you want to obtain all stylize and generator filters, you must call the `filterNamesInCategories:` method for each category separately and then merge the results.

Availability
Mac OS X v10.4 and later.

See Also
`+ filterNamesInCategory:` (page 8)

Related Sample Code
CIAnnotation
CITransitionSelectorSample2

Declared In
CIFilter.h

filterNamesInCategory:

Returns an array of all published filter names in the specified category.

```
+ (NSArray *)filterNamesInCategory:(NSString *)category
```

Parameters

category

A string object that specifies a filter category.

Return Value
An array that contains all published names of the filter in a category.

Availability
Mac OS X v10.4 and later.

See Also
+ filterNamesInCategories: (page 7)

Declared In
CIFilter.h

filterWithImageData:options:

Returns a CIFilter object initialized with RAW image data supplied to the method.

```
+ (CIFilter *)filterWithImageData:(NSData *)data options:(NSDictionary *)options;
```

Parameters

data

The RAW image data to initialize the object with.

options

A options dictionary. You can pass any of the keys defined in "RAW Image Options" (page 29) along with the appropriate value. You should provide a source type identifier hint key (kCGImageSourceTypeIdentifierHint) and the appropriate source type value to help the decoder determine the file type. Otherwise it's possible to obtain incorrect results. See the Discussion for an example

Return Value
A CIFilter object.

Discussion
After calling this method, the CIFilter object returns a CIImage object that is properly processed similar to images retrieved using the outputImage key.

Here is an example of adding a source type identifier key-value pair to the options dictionary:

```
[opts setObject:(id)CGImageSourceGetTypeWithExtension ((CFStringRef)[[url path]
 pathExtension])
         forKey:(id)kCGImageSourceTypeIdentifierHint];
```

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterWithImageURL:options:` (page 9)

Declared In
`CIRAWFilter.h`

filterWithImageURL:options:

Returns a `CIFilter` object initialized with data from a RAW image file.

```
+ (CIFilter *)filterWithImageURL:(NSURL *)url options:(NSDictionary *)options;
```

Parameters

url

The location of a RAW image file.

options

An options dictionary. You can pass any of the keys defined in "`RAW Image Options`" (page 29).

Return Value
A `CIFilter` object.

Discussion
After calling this method, the `CIFilter` object returns a `CIImage` object that is properly processed similar to images retrieved using the `outputImage` key.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterWithImageData:options:` (page 8)

Declared In
`CIRAWFilter.h`

filterWithName:

Creates a `CIFilter` object for a specific kind of filter.

```
+ (CIFilter *)filterWithName:(NSString *)name
```

Parameters

name

The name of the filter.

Return Value
A `CIFilter` object whose input values are undefined.

Discussion
You should call `setDefaults` (page 16) after you call this method or set values individually by calling `setValue:forKey`.

Availability
Mac OS X v10.4 and later.

See Also
`+ filterWithName:keysAndValues:` (page 10)

Related Sample Code
CarbonCocoaCoreImageTab
CIAnnotation
CIVideoDemoGL
QTCoreImage101
Reducer

Declared In
`CIFilter.h`

filterWithName:keysAndValues:

Creates a `CIFilter` object for a specific kind of filter and initializes the input values.

```
+ (CIFilter *)filterWithName:(NSString *)namekeysAndValues:key0, ...
```

Parameters

name

The name of the filter.

key0

A list of key-value pairs to set as input values to the filter. Each key is a constant that specifies the name of the input value to set, and must be followed by a value. You signal the end of the list by passing a `nil` value.

Return Value
A `CIFilter` object whose input values are initialized.

Availability
Mac OS X v10.4 and later.

See Also
`+ filterWithName:` (page 9)

Related Sample Code
CIAnnotation
CITransitionSelectorSample2

Declared In
`CIFilter.h`

localizedDescriptionForFilterName:

Returns the localized description of a filter for display in the user interface.

```
+ (NSString *)localizedDescriptionForFilterName:(NSString *)filterName
```

Parameters

filterName

The filter name.

Return Value

The localized description of the filter.

Availability

Available in Mac OS X v10.5 and later.

Declared In

CIFilter.h

localizedNameForCategory:

Returns the localized name for the specified filter category.

```
+ (NSString *)localizedNameForCategory:(NSString *)category
```

Parameters

category

A filter category.

Return Value

The localized name for the filter category.

Availability

Mac OS X v10.4 and later.

Declared In

CIFilter.h

localizedNameForFilterName:

Returns the localized name for the specified filter name.

```
+ (NSString *)localizedNameForFilterName:(NSString *)filterName
```

Parameters

filterName

A filter name.

Return Value

The localized name for the filter.

Availability

Mac OS X v10.4 and later.

Related Sample Code

QTRecorder

Declared In

CIFilter.h

localizedReferenceDocumentationForFilterName:

Returns the location of the localized reference documentation that describes the filter.

```
+ (NSURL *)localizedReferenceDocumentationForFilterName:(NSString *)filterName
```

Parameters

filterName

The filter name.

Return Value

A URL that specifies the location of the localized documentation, or `nil` if the filter does not provide localized reference documentation.

Discussion

The URL can be a local file or a remote document on a web server. Because filters created prior to Mac OS X v10.5 could return `nil`, you should be make sure that your code handles this case gracefully.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CIFilter.h`

registerFilterName:constructor:classAttributes:

Publishes a custom filter that is not packaged as an image unit.

```
+ (void)registerFilterName:(NSString *)name constructor:(id)anObject
    classAttributes:(NSDictionary *)attributes
```

Parameters

name

A string object that specifies the name of the filter you want to publish.

anObject

A constructor object that implements the `filterWithName` method.

attributes

A dictionary that contains the class display name and filter categories attributes along with the appropriate value for each attributes. That is, the `kCIAttributeFilterDisplayName` (page 17) attribute and a string that specifies the display name, and the `kCIAttributeFilterCategories` (page 17) and an array that specifies the categories to which the filter belongs (such as `kCICategoryStillImage` (page 23) and `kCICategoryDistortionEffect` (page 21)). All other attributes for the filter should be returned by the custom `attributes` method implement by the filter.

Discussion

In most cases you don't need to use this method because the preferred way to register a custom filter that you write is to package it as an image unit. You do not need to use this method for a filter packaged as an image unit because you register your filter using the `CIPlugInRegistration` protocol. (See *Core Image Programming Guide* for additional details.)

Availability

Mac OS X v10.4 and later.

Declared In
`CIFilter.h`

Instance Methods

apply:

Produces a `CIImage` object by applying a kernel function.

```
- (CIImage *)apply:(CIKernel *)k, ...
```

Parameters

k

A `CIKernel` object that contains a kernel function.

A list of arguments to supply to the kernel function. The supplied arguments must be type-compatible with the function signature of the kernel function. The list of arguments must be terminated by the `nil` object.

Discussion
For example, if the kernel function has this signature:

```
kernel vec4 brightenEffect (sampler src, float k)
```

You would supply two arguments after the `k` argument to the `apply:k, ..` method. In this case, the first argument must be a sampler and the second a floating-point value. For more information on kernels, see *Core Image Kernel Language Reference*.

Availability
Mac OS X v10.4 and later.

See Also
`- apply:arguments:options:` (page 13)

Declared In
`CIFilter.h`

apply:arguments:options:

Produces a *CIImage* object by applying arguments to a kernel function and using options to control how the kernel function is evaluated.

```
- (CIImage *)apply:(CIKernel *)k arguments:(NSArray *)args options:(NSDictionary
    *)dict
```

Parameters

k

A `CIKernel` object that contains a kernel function.

args

The arguments that are type compatible with the function signature of the kernel function.

dict

A dictionary that contains options (key-value pairs) to control how the kernel function is evaluated.

Return Value
The `CIImage` object produced by a filter.

Discussion
You can pass any of the following keys in the dictionary:

- `kCIApplyOptionExtent` specifies the size of the produced image. The associated value is a four-element array (`NSArray`) that specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width, and height.
- `kCIApplyOptionDefinition` specifies the domain of definition (DOD) of the produces image. The associated value is either a Core Image filter shape or a four-element array (`NSArray`) that specifies a rectangle.
- `kCIApplyOptionUserInfo` specifies to retain the associated object and pass it to any callbacks invoked for that filter.

Availability
Mac OS X v10.4 and later.

See Also
`- apply:` (page 13)

Declared In
`CIFilter.h`

attributes

Returns a dictionary of key-value pairs that describe the filter.

```
- (NSDictionary *)attributes
```

Return Value
A dictionary that contains a key for each input and output parameter for the filter. Each key is a dictionary that contains all the attributes of an input or output parameter.

Discussion
For example, the attributes dictionary for the `CIColorControls` filter contains the following information:

```
CIColorControls:
{
    CIAttributeFilterCategories = (
        CICategoryColorAdjustment,
        CICategoryVideo,
        CICategoryStillImage,
        CICategoryInterlaced,
        CICategoryNonSquarePixels,
        CICategoryBuiltIn
    );
    CIAttributeFilterDisplayName = "Color Controls";
    CIAttributeFilterName = CIColorControls;
    inputBrightness = {
        CIAttributeClass = NSNumber;
```

```
        CIAttributeDefault = 0;
        CIAttributeIdentity = 0;
        CIAttributeMin = -1;
        CIAttributeSliderMax = 1;
        CIAttributeSliderMin = -1;
        CIAttributeType = CIAttributeTypeScalar;
    };
    inputContrast = {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 1;
        CIAttributeIdentity = 1;
        CIAttributeMin = 0.25;
        CIAttributeSliderMax = 4;
        CIAttributeSliderMin = 0.25;
        CIAttributeType = CIAttributeTypeScalar;
    };
    inputImage = {CIAttributeClass = CIImage; };
    inputSaturation = {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 1;
        CIAttributeIdentity = 1;
        CIAttributeMin = 0;
        CIAttributeSliderMax = 3;
        CIAttributeSliderMin = 0;
        CIAttributeType = CIAttributeTypeScalar;
    };
    outputImage = {CIAttributeClass = CIImage; };
}
```

Availability
Mac OS X v10.4 and later.

Related Sample Code
CITransitionSelectorSample2

Declared In
`CIFilter.h`

inputKeys

Returns an array that contains the names of the input parameters to the filter.

```
- (NSArray *)inputKeys
```

Return Value
An array that contains the names of all input parameters to the filter.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CITransitionSelectorSample2

Declared In
`CIFilter.h`

outputKeys

Returns an array that contains the names of the output parameters for the filter.

```
- (NSArray *)outputKeys
```

Return Value
An array that contains the names of all output parameters from the filter.

Availability
Mac OS X v10.4 and later.

Declared In
`CIFilter.h`

setDefaults

Sets all input values for a filter to default values.

```
- (void)setDefaults
```

Discussion
Input values whose default values are not defined are left unchanged.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CarbonCocoaCoreImageTab
Core Animation QuickTime Layer
QTCarbonCoreImage101
QTRecorder
UnsharpMask

Declared In
`CIFilter.h`

Constants

Filter Attribute Keys

Attributes for a filter and its parameters.

```
extern NSString *kCIAttributeFilterName;
extern NSString *kCIAttributeFilterDisplayName;
extern NSString *kCIAttributeDescription;
extern NSString *kCIAttributeReferenceDocumentation;
extern NSString *kCIAttributeFilterCategories;
extern NSString *kCIAttributeClass;
extern NSString *kCIAttributeType;
extern NSString *kCIAttributeMin;
extern NSString *kCIAttributeMax;
extern NSString *kCIAttributeSliderMin;
extern NSString *kCIAttributeSliderMax;
extern NSString *kCIAttributeDefault;
extern NSString *kCIAttributeIdentity;
extern NSString *kCIAttributeName;
extern NSString *kCIAttributeDisplayName;
```

Constants

`kCIAttributeFilterName`

The filter name, specified as an `NSString` object.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeFilterDisplayName`

The localized version of the filter name that is displayed in the user interface.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeDescription`

The localized description of the filter. This description should inform the end user what the filter does and be short enough to display in the user interface for the filter. It is not intended to be technically detailed.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIAttributeReferenceDocumentation`

The localized reference documentation for the filter. The reference should provide developers with technical details.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIAttributeFilterCategories`

An array of filter category keys that specifies all the categories in which the filter is a member.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeClass`

The class of the input parameter for a filter. If you are writing an image unit (see *Image Unit Tutorial*), Core Image supports only these classes for nonexecutable image units: `CIColor`, `CIVector`, `CIImage`, and `NSNumber` only. Executable image units may have input parameters of any class, but Core Image does not generate an automatic user interface for custom classes (see `CIFilter(IKFilterUIAddition)`).

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeType`
The attribute type.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeMin`
The minimum value for a filter parameter, specified as a floating-point value.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeMax`
The maximum value for a filter parameter, specified as a floating-point value.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeSliderMin`
The minimum value, specified as a floating-point value, to use for a slider that controls input values for a filter parameter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeSliderMax`
The maximum value, specified as a floating-point value, to use for a slider that controls input values for a filter parameter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeDefault`
The default value, specified as a floating-point value, for a filter parameter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeIdentity`
If supplied as a value for a parameter, the parameter has no effect on the input image.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeName`
The name of the attribute.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeDisplayName`
The localized display name of the attribute.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Discussion
Attribute keys are used for the attribute dictionary of a filter. Most entries in the attribute dictionary are optional. The attribute `CIAttributeFilterName` is mandatory. For a parameter, the attribute `kCIAttributeClass` is mandatory.

A parameter of type `NSNumber` does not necessarily need the attributes `kCIAttributeMin` and `kCIAttributeMax`. These attributes are not present when the parameter has no upper or lower bounds. For example, the Gaussian blur filter has a radius parameter with a minimum of `0` but no maximum value to indicate that all nonnegative values are valid.

Declared In
`CIFilter.h`

Data Type Attributes

Numeric data types.

```
extern NSString *kCIAttributeTypeTime;
extern NSString *kCIAttributeTypeScalar;
extern NSString *kCIAttributeTypeDistance;
extern NSString *kCIAttributeTypeAngle;
extern NSString *kCIAttributeTypeBoolean;
extern NSString *kCIAttributeTypeInteger;
extern NSString *kCIAttributeTypeCount;
```

Constants

`kCIAttributeTypeTime`
A parametric time for transitions, specified as a floating-point value in the range of `0.0` to `1.0`.
Available in Mac OS X v10.4 and later.
Declared in `CIFilter.h`.

`kCIAttributeTypeScalar`
A scalar value.
Available in Mac OS X v10.4 and later.
Declared in `CIFilter.h`.

`kCIAttributeTypeDistance`
A distance.
Available in Mac OS X v10.4 and later.
Declared in `CIFilter.h`.

`kCIAttributeTypeAngle`
An angle.
Available in Mac OS X v10.4 and later.
Declared in `CIFilter.h`.

`kCIAttributeTypeBoolean`
A Boolean value.
Available in Mac OS X v10.4 and later.
Declared in `CIFilter.h`.

`kCIAttributeTypeInteger`
An integer value.
Available in Mac OS X v10.5 and later.
Declared in `CIFilter.h`.

`kCIAttributeTypeCount`
A positive integer value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Declared In
`CIFilter.h`

Vector Quantity Attributes

Vector data types.

```
extern NSString *kCIAttributeTypePosition;
extern NSString *kCIAttributeTypeOffset;
extern NSString *kCIAttributeTypePosition3;
extern NSString *kCIAttributeTypeRectangle
```

Constants
`kCIAttributeTypePosition`
A two-dimensional location in the working coordinate space. (A 2-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeOffset`
An offset. (A 2-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypePosition3`
A three-dimensional location in the working coordinate space. (A 3-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeRectangle`
A Core Image vector that specifies the *x* and *y* values of the rectangle origin, and the width (*w*) and height (*h*) of the rectangle. The vector takes the form [*x*, *y*, *w*, *h*]. (A 4-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Declared In
`CIFilter.h`

Color Attribute Keys

Color types.

```
extern NSString *kCIAttributeTypeOpaqueColor;
extern NSString *kCIAttributeTypeGradient;
```

Constants

`kCIAttributeTypeOpaqueColor`

A Core Image color (`CIColor` object) that specifies red, green, and blue component values. Use this key for colors with no alpha component. If the key is not present, Core Image assumes color with alpha.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeGradient`

An n-by-1 gradient image used to describe a color ramp.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Declared In

`CIFilter.h`

Filter Category Keys

Categories of filters.

```
extern NSString *kCICategoryDistortionEffect;
extern NSString *kCICategoryGeometryAdjustment;
extern NSString *kCICategoryCompositeOperation;
extern NSString *kCICategoryHalftoneEffect;
extern NSString *kCICategoryColorAdjustment;
extern NSString *kCICategoryColorEffect;
extern NSString *kCICategoryTransition;
extern NSString *kCICategoryTileEffect;
extern NSString *kCICategoryGenerator;
extern NSString *kCICategoryReduction;
extern NSString *kCICategoryGradient;
extern NSString *kCICategoryStylize;
extern NSString *kCICategorySharpen;
extern NSString *kCICategoryBlur;
extern NSString *kCICategoryVideo;
extern NSString *kCICategoryStillImage;
extern NSString *kCICategoryInterlaced;
extern NSString *kCICategoryNonSquarePixels;
extern NSString *kCICategoryHighDynamicRange ;
extern NSString *kCICategoryBuiltIn;
extern NSString *kCICategoryFilterGenerator;
```

Constants

`kCICategoryDistortionEffect`

A filter that reshapes an image by altering its geometry to create a 3D effect. Using distortion filters, you can displace portions of an image, apply lens effects, make a bulge in an image, and perform other operation to achieve an artistic effect.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryGeometryAdjustment`

A filter that changes the geometry of an image. Some of these filters are used to warp an image to achieve an artistic effects, but these filters can also be used to correct problems in the source image. For example, you can apply an affine transform to straighten an image that is rotated with respect to the horizon.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryCompositeOperation`

A filter operates on two image sources, using the color values of one image to operate on the other. Composite filters perform computations such as computing maximum values, minimum values, and multiplying values between input images. You can use compositing filters to add effects to an image, crop an image, and achieve a variety of other effects.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryHalftoneEffect`

A filter that simulates a variety of halftone screens, to mimic the halftone process used in print media. The output of these filters has the familiar "newspaper" look of the various dot patterns. Filters are typically named after the pattern created by the virtual halftone screen, such as circular screen or hatched screen.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryColorAdjustment`

A filter that changes color values. Color adjustment filters are used to eliminate color casts, adjust hue, and correct brightness and contrast. Color adjustment filters do not perform color management; ColorSync performs color management. You can use Quartz 2D to specify the color space associated with an image. For more information, see *Color Management Overview* and *Quartz 2D Programming Guide*.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryColorEffect`

A filter that modifies the color of an image to achieve an artistic effect. Examples of color effect filters include filters that change a color image to a sepia image or a monochrome image or that produces such effects as posterizing.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryTransition`

A filter that provides a bridge between two or more images by applying a motion effect that defines how the pixels of a source image yield to that of the destination image.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryTileEffect`

A filter that typically applies an effect to an image and then create smaller versions of the image (tiles), which are then laid out to create a pattern that's infinite in extent.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryGenerator`
: A filter that generates a pattern, such as a solid color, a checkerboard, or a star shine. The generated output is typically used as input to another filter.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryReduction`
: A filter that reduces image data. These filters are used to solve image analysis problems.

  Available in Mac OS X v10.5 and later.

  Declared in `CIFilter.h`.

`kCICategoryGradient`
: A filter that generates a fill whose color varies smoothly. Exactly how color varies depends on the type of gradient—linear, radial, or Gaussian.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryStylize`
: A filter that makes a photographic image look as if it was painted or sketched. These filters are typically used alone or in combination with other filters to achieve artistic effects.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategorySharpen`
: A filter that sharpens images, increasing the contrast between the edges in an image. Examples of sharpen filters are unsharp mask and sharpen luminance.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryBlur`
: A filter that softens images, decreasing the contrast between the edges in an image. Examples of blur filters are Gaussian blur and zoom blur.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryVideo`
: A filter that works on video images.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryStillImage`
: A filter that works on still images.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryInterlaced`
: A filter that works on interlaced images.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCICategoryNonSquarePixels`
A filter that works on non-square pixels.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryHighDynamicRange`
A filter that works on high dynamic range pixels.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryBuiltIn`
A filter provided by Core Image. This distinguishes built-in filters from plug-in filters.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryFilterGenerator`
A filter created by chaining several filters together and then packaged as a `CIFilterGenerator` object.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Declared In
`CIFilter.h`

Options for Applying a Filter

Options that control the application of a Core Image filter.

```
extern NSString *kCIApplyOptionExtent;
extern NSString *kCIApplyOptionDefinition;
extern NSString *kCIApplyOptionUserInfo;
```

Constants
`kCIApplyOptionExtent`
The size of the produced image. The associated value is a four-element array (`NSArray`) that specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width and height.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIApplyOptionDefinition`
The domain of definition (DOD) of the produced image. The associated value is either a Core Image filter shape or a four-element array (`NSArray`) that specifies a rectangle.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIApplyOptionUserInfo`
Information needed by a callback. The associated value is an object that Core Image will pass to any callbacks invoked for that filter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Declared In
`CIFilter.h`

User Interface Control Options

Sets of controls for various user scenarios.

```
extern NSString *kCIUIParameterSet;
extern NSString *kCIUISetBasic;
extern NSString *kCIUISetIntermediate;
extern NSString *kCIUISetAdvanced;
extern NSString *kCIUISetDevelopment;
```

Constants

`kCIUIParameterSet`

The set of input parameters to use. The associated value can be `kCIUISetBasic` (page 25), `kCIUISetIntermediate` (page 25), `kCIUISetAdvanced` (page 25), or `kCIUISetDevelopment` (page 25).

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetBasic`

Controls that are appropriate for a basic user scenario, that is, the minimum of settings to control the filter.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetIntermediate`

Controls that are appropriate for an intermediate user scenario.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetAdvanced`

Controls that are appropriate for an advanced user scenario.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetDevelopment`

Controls that should be visible only for development purposes.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Discussion

You can use these constants to specify the controls that you want associated with each user scenario. For example, for a filter that has many input parameters you can choose a small set of input parameters that the typical consumer can control and set the other input parameters to default values. For the same filter, however, you can choose to allow professional customers to control all the input parameters.

Declared In

`CIFIlter.h`

Filter Parameter Keys

Keys for input parameters to filters.

```
extern NSString *kCIOutputImageKey;
extern NSString *kCIInputBackgroundImageKey;
extern NSString *kCIInputImageKey;
extern NSString *kCIInputTimeKey;
extern NSString *kCIInputTransformKey;
extern NSString *kCIInputScaleKey;
extern NSString *kCIInputAspectRatioKey;
extern NSString *kCIInputCenterKey;
extern NSString *kCIInputRadiusKey;
extern NSString *kCIInputAngleKey;
extern NSString *kCIInputRefractionKey;
extern NSString *kCIInputWidthKey;
extern NSString *kCIInputSharpnessKey;
extern NSString *kCIInputIntensityKey;
extern NSString *kCIInputEVKey;
extern NSString *kCIInputSaturationKey;
extern NSString *kCIInputColorKey;
extern NSString *kCIInputBrightnessKey;
extern NSString *kCIInputContrastKey;
extern NSString *kCIInputGradientImageKey;
extern NSString *kCIInputMaskImageKey;
extern NSString *kCIInputShadingImageKey;
extern NSString *kCIInputTargetImageKey;
extern NSString *kCIInputExtentKey;
```

Constants

`kCIOutputImageKey`

A key for the `CIImage` object produced by a filter.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputBackgroundImageKey`

A key for the `CIImage` object to use as a background image.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputImageKey`

A key for the `CIImage` object to use as an input image. For filters that also use a background image, this key refers to the foreground image.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputTimeKey`

A key for z scalar value (`NSNumber`) that specifies a time.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputTransformKey`

A key for an `NSAffineTransform` object that specifies a transformation to apply.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputScaleKey`
A key for a scalar value (`NSNumber`) that specifies the amount of the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputAspectRatioKey`
A key for a scalar value (`NSNumber`) that specifies a ratio.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputCenterKey`
A key for a `CIVector` object that specifies the center of the area, as *x* and *y*- coordinates, to be filtered.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputRadiusKey`
A key for a scalar value (`NSNumber`) that specifies that specifies the distance from the center of an effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputAngleKey`
A key for a scalar value (`NSNumber`) that specifies an angle.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputRefractionKey`
A key for a scalar value (`NSNumber`) that specifies the index of refraction of the material (such as glass) used in the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputWidthKey`
A key for a scalar value (`NSNumber`) that specifies the width of the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputSharpnessKey`
A key for a scalar value (`NSNumber`) that specifies the amount of sharpening to apply.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputIntensityKey`
A key for a scalar value (`NSNumber`) that specifies an intensity value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputEVKey`
A key for a scalar value (`NSNumber`) that specifies how many F-stops brighter or darker the image should be.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputSaturationKey`
A key for a scalar value (`NSNumber`) that specifies the amount to adjust the saturation.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputColorKey`
A key for a `CIColor` object that specifies a color value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputBrightnessKey`
A key for a scalar value (`NSNumber`) that specifies a brightness level.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputContrastKey`
A key for a scalar value (`NSNumber`) that specifies a contrast level.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputGradientImageKey`
A key for a `CIImage` object that specifies an environment map with alpha. Typically, this image contains highlight and shadow.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputMaskImageKey`
A key for a `CIImage` object to use as a mask.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputShadingImageKey`
A key for a `CIImage` object that specifies an environment map with alpha values. Typically this image contains highlight and shadow.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputTargetImageKey`
A key for a `CIImage` object that is the target image for a transition.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputExtentKey`
A key for a `CIVector` object that specifies a rectangle that defines the extent of the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Discussion
These keys represent some of the most commonly used input parameters. A filter can use other kinds of input parameters.

Declared In
`CIFIlter.h`

RAW Image Options

Options for creating a `CIFilter` object from RAW image data.

```
extern NSString * const kCIInputDecoderVersionKey;
extern NSString * const kCISupportedDecoderVersionsKey;
extern NSString * const kCIInputBoostKey;
extern NSString * const kCIInputNeutralChromaticityXKey;
extern NSString * const kCIInputNeutralChromaticityYKey;
extern NSString * const kCIInputNeutralTemperatureKey;
extern NSString * const kCIInputNeutralTintKey;
extern NSString * const kCIInputNeutralLocation;
extern NSString * const kCIInputScaleFactorKey;
extern NSString * const kCIInputAllowDraftModeKey;
extern NSString * const kCIInputIgnoreImageOrientationKey;
extern NSString * const kCIInputImageOrientationKey;
extern NSString * const kCIInputEnableSharpeningKey;
extern NSString * const kCIInputEnableChromaticNoiseTrackingKey;
extern NSString * const kCIInputBoostShadowAmountKey;
extern NSString * const kCIInputBiasKey;
```

Constants

`kCIInputDecoderVersionKey`

A key for the version number of the method to be used for decoding. A newly initialized object defaults to the newest available decoder version for the given image type. You can request an alternative, older version to maintain compatibility with older releases. Must be one of `kCISupportedDecoderVersions`, otherwise a `nil` output image is generated. The associated value must be an `NSNumber` object that specifies an integer value in range of `0` to the current decoder version. When you request a specific version of the decoder, Core Image produces an image that is *visually* the same across different versions of the operating system. Core Image, however, does not guarantee that the same bits are produced across different versions of the operating system. That's because the rounding behavior of floating-point arithmetic can vary due to differences in compilers or hardware. Note that this option has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCISupportedDecoderVersionsKey`

A key for the supported decoder versions. The associated value is an `NSArray` object that contains all supported decoder versions for the given image type, sorted in increasingly newer order. Each entry is an `NSDictionary` object that contains key-value pairs. All entries represent a valid version identifier that can be passed as the `kCIDecoderVersion` value for the key `kCIDecoderMethodKey`. Version values are read-only; attempting to set this value raises an exception. Currently, the only defined key is `@"version"` which has as its value an `NSString` that uniquely describing a given decoder version. This string might not be suitable for user interface display..

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputBoostKey`

A key for the the amount of boost to apply to an image. The associated value is a floating-point value packaged as an `NSNumber` object. The value must be in the range of `0...1`. A value of `0` indicates no boost, that is, a linear response. The default value is `1`, which indicates full boost.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralChromaticityXKey`

The x value of the chromaticity. The associated value is a floating-point value packaged as an `NSNumber` object. You can query this value to get the current x value for neutral x, y.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralChromaticityYKey`

The y value of the chromaticity. The associated value is a floating-point value packaged as an `NSNumber` object. You can query this value to get the current y value for neutral x, y.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralTemperatureKey`

A key for neutral temperature. The associated value is a floating-point value packaged as an `NSNumber` object. You can query this value to get the current temperature value.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralTintKey`

A key for the neutral tint. The associated value is a floating-point value packaged as an `NSNumber` object. Use this key to set or fetch the temperature and tint values. You can query this value to get the current tint value.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralLocationKey`

A key for the neutral position. Use this key to set the location in geometric coordinates of the unrotated output image that should be used as neutral. You cannot query this value; it is undefined for reading. The associated value is a two-element `CIVector` object that specifies the location (`x`, `y`).

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputScaleFactorKey`

A key for the scale factor. The associated value is a floating-point value packaged as an `NSNumber` object that specifies the desired scale factor at which the image will be drawn. Setting this value can greatly improve the drawing performance. A value of `1` is the identity. In some cases, if you change the scale factor and enable draft mode, performance can decrease. See `kCIAllowDraftModeKey`.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputAllowDraftModeKey`

A key for allowing draft mode. The associated value is a Boolean value packaged as an `NSNumber` object. It's best not to use draft mode if the image needs to be drawn without draft mode at a later time, because changing the value from `YES` to `NO` is an expensive operation. If the optional scale factor is smaller than a certain value, additionally setting draft mode can improve image decoding speed without any perceivable loss of quality. However, turning on draft mode does not have any effect if the scale factor is not below this threshold.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputIgnoreImageOrientationKey`

A key for specifying whether to ignore the image orientation. The associated value is a Boolean value packaged as an `NSNumber` object. The default value is `NO`. An image is usually loaded in its proper orientation, as long as the associated metadata records its orientation. For special purposes you might want to load the image in its physical orientation. The exact meaning of "physical orientation" is dependent on the specific image.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputImageOrientationKey`

A key for the image orientation. The associated value is an integer value packaged as an `NSNumber` object. Valid values are in range `1...8` and follow the EXIF specification. The value is disregarded when the `kCIIgnoreImageOrientationKey` flag is set. You can change the orientation of the image by overriding this value. By changing this value you can easily rotate an image in 90-degree increments.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputEnableSharpeningKey`

A key for the sharpening state. The associated value must be an `NSNumber` object that specifies a `BOOL` value (`YES` or `NO`). The default is `YES`. This option has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputEnableChromaticNoiseTrackingKey`

A key for progressive chromatic noise tracking (based on ISO and exposure time). The associated value must be an `NSNumber` object that specifies a `BOOL` value (`YES` or `NO`). The default is `YES`. This option has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputBoostShadowAmountKey`

A key for the amount to boost the shadow areas of the image. The associated value must be an `NSNumber` object that specifies floating-point value. The value has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputBiasKey`

A key for the simple bias value to use along with the exposure adjustment (`kCIInputEVKey`). The associated value must be an `NSNumber` object that specifies floating-point value. The value has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

Discussion

You can also use the key `kCIInputEVKey` for RAW images.

Declared In

`CIRAWFilter.h`

Document Revision History

This table describes the changes to *CIFilter Class Reference*.

Date	Notes
2007-12-11	Added a filter category and updated filter attribute constants.
	See `kCIAttributeClass` (page 17) and "`Filter Category Keys`" (page 21).
2007-06-26	Updated for Mac OS X v10.5.
	Added additions to support using RAW images. See "Creating a Filter from a RAW Image" (page 6) and "`RAW Image Options`" (page 29).
	Added the methods `localizedDescriptionForFilterName:` (page 10) and `localizedReferenceDocumentationForFilterName:` (page 12).
	Add the constant groups: "`User Interface Control Options`" (page 25) and "`Filter Parameter Keys`" (page 25).
	Added two filter attributes: `kCIAttributeDescription` (page 17) and `kCIAttributeReferenceDocumentation` (page 17).
	Added two attribute types: `kCIAttributeTypeInteger` (page 19) and `kCIAttributeTypeCount` (page 20).
	Changed formatting for constants.
2006-06-28	Clarified a few technical points.
	Added a discussion to `registerFilterName:constructor:classAttributes:` (page 12).
2006-05-23	First publication of this content as a separate document.
	Added parameter descriptions and updated Class Description.

Document Revision History

Index

A

apply: instance method 13
apply:arguments:options: instance method 13
attributes instance method 14

C

Color Attribute Keys 20

D

Data Type Attributes 19

F

Filter Attribute Keys 16
Filter Category Keys 21
Filter Parameter Keys 25
filterNamesInCategories: class method 7
filterNamesInCategory: class method 8
filterWithImageData:options: class method 8
filterWithImageURL:options: class method 9
filterWithName: class method 9
filterWithName:keysAndValues: class method 10

I

inputKeys instance method 15

K

kCIApplyOptionDefinition constant 24
kCIApplyOptionExtent constant 24
kCIApplyOptionUserInfo constant 24
kCIAttributeClass constant 17
kCIAttributeDefault constant 18
kCIAttributeDescription constant 17
kCIAttributeDisplayName constant 18
kCIAttributeFilterCategories constant 17
kCIAttributeFilterDisplayName constant 17
kCIAttributeFilterName constant 17
kCIAttributeIdentity constant 18
kCIAttributeMax constant 18
kCIAttributeMin constant 18
kCIAttributeName constant 18
kCIAttributeReferenceDocumentation constant 17
kCIAttributeSliderMax constant 18
kCIAttributeSliderMin constant 18
kCIAttributeType constant 18
kCIAttributeTypeAngle constant 19
kCIAttributeTypeBoolean constant 19
kCIAttributeTypeCount constant 20
kCIAttributeTypeDistance constant 19
kCIAttributeTypeGradient constant 21
kCIAttributeTypeInteger constant 19
kCIAttributeTypeOffset constant 20
kCIAttributeTypeOpaqueColor constant 21
kCIAttributeTypePosition constant 20
kCIAttributeTypePosition3 constant 20
kCIAttributeTypeRectangle constant 20
kCIAttributeTypeScalar constant 19
kCIAttributeTypeTime constant 19
kCICategoryBlur constant 23
kCICategoryBuiltIn constant 24
kCICategoryColorAdjustment constant 22
kCICategoryColorEffect constant 22
kCICategoryCompositeOperation constant 22
kCICategoryDistortionEffect constant 21
kCICategoryFilterGenerator constant 24
kCICategoryGenerator constant 23
kCICategoryGeometryAdjustment constant 22
kCICategoryGradient constant 23
kCICategoryHalftoneEffect constant 22
kCICategoryHighDynamicRange constant 24

kCICategoryInterlaced constant 23
kCICategoryNonSquarePixels constant 24
kCICategoryReduction constant 23
kCICategorySharpen constant 23
kCICategoryStillImage constant 23
kCICategoryStylize constant 23
kCICategoryTileEffect constant 22
kCICategoryTransition constant 22
kCICategoryVideo constant 23
kCIInputAllowDraftModeKey constant 30
kCIInputAngleKey constant 27
kCIInputAspectRatioKey constant 27
kCIInputBackgroundImageKey constant 26
kCIInputBiasKey constant 31
kCIInputBoostKey constant 29
kCIInputBoostShadowAmountKey constant 31
kCIInputBrightnessKey constant 28
kCIInputCenterKey constant 27
kCIInputColorKey constant 28
kCIInputContrastKey constant 28
kCIInputDecoderVersionKey constant 29
kCIInputEnableChromaticNoiseTrackingKey constant 31
kCIInputEnableSharpeningKey constant 31
kCIInputEVKey constant 27
kCIInputExtentKey constant 28
kCIInputGradientImageKey constant 28
kCIInputIgnoreImageOrientationKey constant 31
kCIInputImageKey constant 26
kCIInputImageOrientationKey constant 31
kCIInputIntensityKey constant 27
kCIInputMaskImageKey constant 28
kCIInputNeutralChromaticityXKey constant 30
kCIInputNeutralChromaticityYKey constant 30
kCIInputNeutralLocationKey constant 30
kCIInputNeutralTemperatureKey constant 30
kCIInputNeutralTintKey constant 30
kCIInputRadiusKey constant 27
kCIInputRefractionKey constant 27
kCIInputSaturationKey constant 28
kCIInputScaleFactorKey constant 30
kCIInputScaleKey constant 27
kCIInputShadingImageKey constant 28
kCIInputSharpnessKey constant 27
kCIInputTargetImageKey constant 28
kCIInputTimeKey constant 26
kCIInputTransformKey constant 26
kCIInputWidthKey constant 27
kCIOutputImageKey constant 26
kCISupportedDecoderVersionsKey constant 29
kCIUIParameterSet constant 25
kCIUISetAdvanced constant 25
kCIUISetBasic constant 25
kCIUISetDevelopment constant 25
kCIUISetIntermediate constant 25

L

localizedDescriptionForFilterName: class method 10
localizedNameForCategory: class method 11
localizedNameForFilterName: class method 11
localizedReferenceDocumentationForFilterName: class method 12

O

Options for Applying a Filter 24
outputKeys instance method 16

R

RAW Image Options 29
registerFilterName:constructor:classAttributes: class method 12

S

setDefaults instance method 16

U

User Interface Control Options 25

V

Vector Quantity Attributes 20