Export Plugin API

Header files:

```
ExportPluginBoxProtocol.h
ExportPluginProtocol.h
ExportImageProtocol.h
```

ExportPluginBoxProtocol Reference

`ExportPluginBoxProtocol.h`

Introduction

The `ExportPluginBoxProtocol` define methods that must be implemented by a subclass of `NSBox` and will serve as the super view for all plugin views.

Methods by Task

Handling events and action messages

    - performKeyEquivalent:

Methods Details

performKeyEquivalent:

This method is supposed to check the event against the “enter” key and perform the export if they match.

```
- (BOOL)performKeyEquivalent:(NSEvent *)anEvent;
```

Parameters

anEvent

The key-down event object representing a key equivalent.

Return Value

`YES` if *anEvent* is a key equivalent that the receiver handled; `NO` if it is not a key equivalent that it should handle.

Discussion

This method will override the inherited method from `NSView`. If *anEvent* is the “enter” key, it should perform the appropriate action and return `YES`. Otherwise invoke super’s implementation.

ExportPluginProtocol Reference

`ExportPluginProtocol.h`

Introduction

The plugin's controller class must adopt the `ExportPluginProtocol`. It is responsible for managing plugin UI and performing the export.

Methods by Task

Creating an object

```
- initWithExportImageObj:
```

Managing views

```
- settingsView
- firstView
- viewWillBeActivated
- viewWillBeDeactivated
```

Plugin settings

```
- name
- requiredFileType
- wantsDestinationPrompt
- getDestinationPath
- defaultFileName
- defaultDirectory
- treatSingleSelectionDifferently
- handlesMovieFiles
```

Performing export

```
- validateUserCreatedPath:
- clickExport
- startExport:
- performExport:
- cancelExport
```

Progress controls

```
- progress
- lockProgress
- unlockProgress
```

Methods Details

initWithExportImageObj:
Initializes the newly allocated plugin controller object with the specified export manager.

```
- (id)initWithExportImageObj:(id <ExportImageProtocol>)obj;
```

Parameters
obj
The export manager that will help the plugin perform the export.

Return Value
An initialized plugin controller object or `nil` if object couldn't be created.

Discussion
This method will be called when iPhoto's export controller initializes the plugin. *obj* will be used to help with exporting.

settingsView
```
- (NSView <ExportPluginBoxProtocol> *)settingsView;
```

Return Value
The subclass of `NSBox` that is the super view of the plugin controls.

firstView
```
- (NSControl *)firstView;
```

Return Value
The first responder when plugin loads.

viewWillBeActivated
Notifies the receiver it is now the "active" view.

```
- (void)viewWillBeActivated;
```

viewWillBeDeactivated
Notifies the receiver it is no longer the "active" view.

```
- (void)viewWillBeActivated;
```

name

```
- (NSString *)name;
```

Return Value
The name of the plugin.

Discussion
The name is used to identify the plugin.

requiredFileType

Specifies the required file type for export.

```
- (NSString *)requiredFileType;
```

Return Value
Required file type for saving. `@””` indicates directory.

Discussion
When user selects the filename to save as, the file will be saved using the extension returned by this method. Examples of file types are “jpg,” “tiff,” and “ps.” Return value of `@””` indicates a directory (the user will only get to choose an export directory name instead of file name). `NSSavePanel`’s `setFileType` method will be called with the file type returned by this method.

wantsDestinationPrompt

Indicates whether the plugin supports user selected destination.

```
- (BOOL)wantsDestinationPrompt;
```

Return Value
`YES` if the plugin lets user pick filename and location to export. `NO` otherwise.

Discussion
Before starting export, iPhoto sends this message. If `YES` is returned, iPhoto will prompt the user for destination to export. Otherwise, the plugin defined directory and filename will be used.

getDestinationPath

`- (NSString *)getDestinationPath;`

Return Value
The plugin defined export destination.

Discussion
If `wantsDestinationPrompt` returns `NO`, iPhoto sends this message to the plugin to ask for the path to export to.

defaultFileName

`- (NSString *)defaultFileName;`

Return Value
The default filename to be displayed on the save panel.

Discussion
If `wantsDestinationPrompt` returns `YES`, iPhoto sends this message to the plugin to ask for the default filename to be displayed in the save panel. This method will only be called if `requiredFileType` returns a non-empty string.

defaultDirectory

`- (NSString *)defaultDirectory;`

Return Value
The initial directory whose contents are displayed when the save panel starts.

Discussion
If `wantsDestinationPrompt` returns `YES`, the directory returned will be the default directory displayed in the save panel. Return value of `nil` will set the default directory as the directory viewed in the last save panel session.

treatSingleSelectionDifferently

Indicates whether the plugin wants to handle a single image differently from multiple images.

```
- (BOOL)treatSingleSelectionDifferently;
```

Return Value

YES if the plugin wants to work a little differently if the user is exporting just a single image. NO otherwise. If YES is returned and only one image is selected, a save panel will be presented to ask for export directory and filename. Otherwise, an open panel will be presented to ask for export directory only. In the rest of this documentation, “save panel” refers to both save panel and open panel.

handlesMovieFiles

Indicates whether the plugin wants to handle movie files.

```
- (BOOL)handlesMovieFiles;
```

Return Value

YES if the plugin can handle movies. NO otherwise.

Discussion

If NO is returned, only image files are passed to the plugin and movie files are ignored.

validateUserCreatedPath:

Indicates whether the plugin is okay with the export path entered by the user.

```
- (BOOL)validateUserCreatedPath:(NSString*)path;
```

Parameters

path

The export path entered by the user.

Return Value

YES if the plugin is fine with the path. NO otherwise.

Discussion

Called if user enters a destination path that doesn’t exist. If YES is returned, export will proceed. Otherwise export will not continue.

clickExport
Handles “enter” key for exporting.

```
- (void)clickExport;
```

Discussion
This method should be called by `performKeyEquivalent:` from `ExportPluginBoxProtocol`. This method should call `clickExport` method of the export manager to start export.

startExport:
Prepares for export to the given path.

```
- (void)startExport:(NSString *)path;
```

Parameters
path
The export path.

Discussion
Called to initiate export. Should handle any last minute preparation such as checking destination directory or initializing any variables. Should call export manager’s `startExport` if everything’s ready, otherwise return without calling export manager’s `startExport`.

performExport:
Performs the export to the given path.

```
- (void)performExport:(NSString *)path;
```

Parameters
path
The export path.

Discussion
iPhoto’s export controller will call this method to actually do the export. *path* is the same as the path passed to `startExport:`. This method will only be called if `startExport:` calls export manager’s `startExport`.

cancelExport
Cancels the export.

```
- (void)cancelExport;
```

Discussion
Called to cancel an export. Called by iPhoto's export controller if user clicks "cancel" during an export.

progress
```
- (ExportPluginProgress *)progress;
```

Return Value
A pointer to an instance of `ExportPluginProgress` struct.

Discussion
Method for retrieving progress of export.

lockProgress
Mutex lock the progress controls.

```
- (void)lockProgress;
```

Discussion
Called to lock the progress struct for editing. Should be called before changing variable of the struct.

unlockProgress
Unlock the progress controls.

```
- (void)unlockProgress;
```

Discussion
Called to unlock the progress struct. Should be called when editing is done.

Definitions

ExportPluginProgress

Controls the progress sheet while the plugin is performing the export.

```
typedef struct
{
    unsigned long   currentItem;
    unsigned long   totalItems;
    NSString        *message;
    BOOL            indeterminateProgress;
    BOOL            shouldCancel;
    BOOL            shouldStop;
} ExportPluginProgress;
```

Variables

currentItem

The current item number in progress.

totalItems

The total number of items to process.

message

The message to be displayed on the progress sheet.

indeterminateProgress

Whether the current progress is an indeterminate progress.

shouldCancel

Whether export should be canceled. Pauses export progress and waits for user to click “cancel”.

shouldStop

Whether export should be stopped. Closes the export window.

Discussion

This struct controls the look of the progress sheet while the plugin is performing the export. Since the struct variables may be changed from multiple threads, a mutex lock should be used before changing any variables. The plugin should send itself `cancelExport` message before setting shouldCancel to `YES`. *shouldStop* should be set to `YES` when export is finished.

ExportImageProtocol Reference

`ExportImageProtocol.h`

Introduction

The export manager passed to the plugin adopts `ExportPluginProtocol`. The plugin can call these methods to help with export.

Methods by Task

Accessing images

```
- imageCount
- imageSizeAtIndex:
- imageFormatAtIndex:
- originalImageFormatAtIndex:
- originalIsRawAtIndex:
- originalIsMovieAtIndex:
- imageTitleAtIndex:
- imageCommentsAtIndex:
- imageRotationAtIndex:
- imagePathAtIndex:
- sourcePathAtIndex:
- thumbnailPathAtIndex:
- imageFileNameAtIndex:
- imageIsPortraitAtIndex:
- imageAspectRatioAtIndex:
- imageFileSizeAtIndex:
- imageDateAtIndex:
- imageRatingAtIndex:
- imageTiffPropertiesAtIndex:
- imageExifPropertiesAtIndex:
- imageKeywordsAtIndex:
- albumsOfImageAtIndex:
```

Accessing albums

```
- albumCount
- albumNameAtIndex:
- albumMusicPathAtIndex:
- albumCommentsAtIndex:
- positionOfImageAtIndex:inAlbum:
```

Accessing export GUI

```
- window
- enableControls
- disableControls
```

Performing export

```
- sessionID
- directoryPath
- clickExport
- startExport
- cancelExportBeforeBeginning
- exportImageAtIndex:dest:options:
- lastExportedImageSize
```

Methods Details

imageCount

```
- (unsigned)imageCount;
```

Return Value
The number of images/movies selected for export.

Discussion
If the plugin does not handle movies, then the count will only include selected images.

imageSizeAtIndex:

```
- (NSSize)imageSizeAtIndex:(unsigned)index;
```

Parameters
index
The index of the image.

Return Value
The size of the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive.

imageFormatAtIndex:

```
- (OSType)imageFormatAtIndex:(unsigned)index;
```

Parameters
index
The index of the image.

Return Value
The format of the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive.

originalImageFormatAtIndex:

```
- (OSType)originalImageFormatAtIndex:(unsigned)index;
```

Parameters
index
The index of the image.

Return Value
The format of the original image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive.
Original image is the image before imported into iPhoto.

originalIsRawAtIndex:

Indicates whether original image is raw.

```
- (BOOL)originalIsRawAtIndex:(unsigned)index;
```

Parameters
index
The index of the image.

Return Value
`YES` if original image is a raw file. `NO` otherwise.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive.
Original image is the image before imported into iPhoto.

originalIsMovieAtIndex:
Indicates whether original file is a movie file.

```
- (BOOL)originalIsMovieAtIndex:(unsigned)index;
```

Parameters
index

The index of the image.

Return Value
`YES` if original file is a movie file. `NO` otherwise.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive. Original image is the image before imported into iPhoto. This method will always return `NO` if `handlesMovieFiles` method of the plugin returns `NO`.

imageTitleAtIndex:
```
- (NSString *)imageTitleAtIndex:(unsigned)index;
```

Parameters
index

The index of the image.

Return Value
The user defined title of the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive. The default user defined title is the filename of the image.

imageCommentsAtIndex:
```
- (NSString *)imageCommentsAtIndex:(unsigned)index;
```

Parameters
index

The index of the image.

Return Value
The user comments of the image.

Discussion

`index` is an unsigned integer between 0 and `imageCount-1` inclusive. If no user comments for the image, an empty string is returned.

imageRotationAtIndex:

```
- (float)imageRotationAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The amount of rotation in degrees that should be applied to the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. This method returns a non-zero float if an image has been flagged to be rotated but has not yet been processed. Positive value is clockwise rotation and negative value is counter clockwise rotation.

imagePathAtIndex:

```
- (NSString *)imagePathAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The path of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. The path points to the most recently edited version of the image. If the original image has not been edited and is not raw, it will return the path of the original image. If the original image is raw, it will return the path to a JPEG version of the original image.

sourcePathAtIndex:

```
- (NSString *)sourcePathAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The source path of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. If original image is raw or is movie, source path will point to the raw/movie file. Otherwise, source path points to the same path as `imagePathAtIndex:`.

thumbnailPathAtIndex:

```
- (NSString *)thumbnailPathAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The path of the large JPEG thumbnail of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. The thumbnail image does not contain any metadata and has a maximum size of 360 by 360 pixels.

imageFileNameAtIndex:

```
- (NSString *)imageFileNameAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The filename of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive.

imageIsPortraitAtIndex:

Indicates whether image is portrait.

```
- (BOOL)imageIsPortraitAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

`YES` if image is portrait. `NO` otherwise.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive.

imageAspectRatioAtIndex:

```
- (float)imageAspectRatioAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The aspect ratio of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. The aspect ratio is image width divided by image height.

imageFileSizeAtIndex:

```
- (unsigned long long)imageFileSizeAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The file size of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive.

imageDateAtIndex:

```
- (NSDate *)imageDateAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The creation date of the image as displayed in the information panel.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. This date may be different from the date in the TIFF metadata (displayed in the floating information window) recorded by the camera.

imageRatingAtIndex:

```
- (int)imageRatingAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

The user rating of the image.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. Rating is an integer between 0 and 5 inclusive.

imageTiffPropertiesAtIndex:

```
- (NSDictionary *)imageTiffPropertiesAtIndex:(unsigned)index;
```

Parameters

index

The index of the image.

Return Value

A dictionary containing TIFF metadata of the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive. See TIFF metadata dictionary access keys. Objects are formatted strings. If a particular data is not available, `@”--”` is returned.

imageExifPropertiesAtIndex:

```
- (NSDictionary *)imageExifPropertiesAtIndex:(unsigned)index;
```

Parameters
index
    The index of the image.

Return Value
A dictionary containing EXIF metadata of the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive. See EXIF metadata dictionary access keys. Objects are formatted strings. If a particular data is not available, `@”--”` is returned.

imageKeywordsAtIndex:

```
- (NSArray *)imageKeywordsAtIndex:(unsigned)index;
```

Parameters
index
    The index of the image.

Return Value
An array containing keywords associated with the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive.

albumsOfImageAtIndex:

```
- (NSArray *)albumsOfImageAtIndex:(unsigned)index;
```

Parameters
index

The index of the image.

Return Value
An array of `NSNumber` objects containing indices of all the selected albums that contains the image.

Discussion
index is an unsigned integer between 0 and `imageCount-1` inclusive.

albumCount

```
- (unsigned)albumCount;
```

Return Value
The number of albums selected.

albumNameAtIndex:

```
- (NSString *)albumNameAtIndex:(unsigned)index;
```

Parameters
index
The index of the album.

Return Value
The name of the album.

Discussion
index is an unsigned integer between 0 and `albumCount-1` inclusive.

albumMusicPathAtIndex:

```
- (NSString *)albumMusicPathAtIndex:(unsigned)index;
```

Parameters
index
The index of the album.

Return Value
The slideshow music path of the album.

Discussion

index is an unsigned integer between 0 and `albumCount-1` inclusive.

albumCommentsAtIndex:

```
- (NSString *)albumCommentsAtIndex:(unsigned)index;
```

Parameters

index

The index of the album.

Return Value

User comments of the album.

Discussion

index is an unsigned integer between 0 and `albumCount-1` inclusive. If no user comments, then empty string is returned.

positionOfImageAtIndex:inAlbum:

```
- (unsigned)positionOfImageAtIndex:(unsigned)index
                           inAlbum:(unsigned)album;
```

Parameters

index

The index of the image.

album

The index of the album.

Return Value

The position of the image in the album. -1 if image is not in album.

Discussion

index is an unsigned integer between 0 and `imageCount-1` inclusive. *album* is an unsigned integer between 0 and `albumCount-1` inclusive. The position of the image includes images in the trashcan.

window

```
- (id)window;
```

Return Value

The export window.

enableControls

Enables the “export” button in the export window.

```
- (void)enableControls;
```

Discussion

Used if `disableControls` disables the “export” button.

disableControls

Disables the “export” button in the export window.

```
- (void)disableControls;
```

Discussion

Used to disabled the “export” button to prevent the user from exporting.

sessionID

The ID of the export session.

```
- (unsigned)sessionID;
```

Discussion

The session ID is an unsigned integer starting from 1 when iPhoto first starts and increments every time the user opens the export panel.

directoryPath

```
- (NSString *)directoryPath;
```

Return Value

The export path. `nil` if path is not set yet.

clickExport

Handles the “enter” key.

```
- (void)clickExport;
```

Discussion
Should be called by the `clickExport` method of the plugin.

startExport
Prepares iPhoto for exporting.

```
- (void)startExport;
```

Discussion
Should be called by the `startExport` method of the plugin.

cancelExportBeforeBeginning
Cancels export before it starts.

```
- (void)cancelExportBeforeBeginning;
```

Discussion
This method causes the export window to close.

exportImageAtIndex:dest:options:
Exports the image to the destination with the options.

```
- (BOOL)exportImageAtIndex:(unsigned)index
                      dest:(NSString *)dest
                   options:(ImageExportOptions *)options;
```

Parameters

index
The index of the image.

dest
The export destination.

options
The options to be used.

Return Value
`YES` if export is successful. `NO` otherwise.

Discussion

dest must not already exist. All variables of the `ImageExportOptions` struct must be set.

lastExportedImageSize

`- (NSSize)lastExportedImageSize;`

Return Value
The resulting image size of the most recently exported image.

Definitions

ImageExportOptions

The options used for exporting image.

```
typedef struct
{
    OSType          format;
    ExportQuality   quality;
    float           rotation;
    unsigned        width;
    unsigned        height;
    ExportMetadata  metadata;
} ImageExportOptions;
```

Variables

format
The image format to export to.

quality
The image compression quality for the export image.

rotation
The amount to rotate the export image in degrees.

width
The maximum width of the export image.

height
The maximum height of the export image.

metadata
Whether to embed EXIF or IPTC metadata in the export image.

Discussion
This struct contains the options used for exporting image. All variables must be set before the struct is used. Image will be rotated by *rotation* mod 360. Positive *rotation* rotates clockwise and negative value rotates counter-clockwise.

ExportQuality

The possible image compression qualities.

```
typedef enum
{
    EQualityLow,
    EQualityMed,
    EQualityHigh,
    EQualityMax
} ExportQuality;
```

Discussion

Low is for the lowest quality and best compression. Max is for the best quality and lowest compression.

ExportMetadata

The possible options for embedding metadata in image.

```
typedef enum
{
    EMNone,
    EMEXIF,
    EMIPTC,
    EMBoth
} ExportMetadata;
```

Discussion

The choices are no metadata, EXIF metadata only, IPTC metadata only, and both EXIF and IPTC metadata.

EXIF metadata access keys

Keys for the dictionary returned by `imageExifPropertiesAtIndex:`.

```
#define kIPExifShutter @"Shutter"
#define kIPExifAperture @"Aperture"
#define kIPExifMaxAperture @"MaxAperture"
#define kIPExifExposureBias @"ExposureBias"
#define kIPExifExposure @"Exposure"
#define kIPExifExposureIndex @"ExposureIndex"
#define kIPExifFocalLength @"FocalLength"
```

```
#define kIPExifDistance @"Distance"
#define kIPExifSensing @"Sensing"
#define kIPExifLightSource @"LightSource"
#define kIPExifFlash @"Flash"
#define kIPExifMetering @"Metering"
#define kIPExifBrightness @"Brightness"
#define kIPExifISOSpeed @"ISOSpeed"
```

TIFF metadata access keys

Keys for the dictionary returned by `imageTiffPropertiesAtIndex:`.

```
#define kIPTiffImageWidth @"ImageWidth"
#define kIPTiffImageHeight @"ImageHeight"
#define kIPTiffOriginalDate @"OriginalDate"
#define kIPTiffDigitizedDate @"DigitizedDate"
#define kIPTiffFileName @"FileName"
#define kIPTiffFileSize @"FileSize"
#define kIPTiffModifiedDate @"ModifiedDate"
#define kIPTiffImportedDate @"ImportedDate"
#define kIPTiffCameraMaker @"CameraMaker"
#define kIPTiffCameraModel @"CameraModel"
#define kIPTiffSoftware @"Software"
```