Converter

INHERITS FROM	Object
DECLARED IN	Converter.h

CLASS DESCRIPTION

The Converter objects provides the mechanism forconverting images between different formats. Each converter object is responsible for reading and writing one graphics format. Through the interaction of many of these object, image conversion is achieved.

In general, you will not need to call these methods directly. This reference is supplied many for people who wish to program their own converters. To that end, the descriptions tend to read more for the converter writer and not the programmer who wishes to use the object.

INSTANCE VARIABLES

Inherited from Object	Class	isa;
Declared in Converter	int	errorState;

METHOD TYPES

Creating and Freeing a Converter	- init - free
Reading and Writing	- readFromStream:from: - readAllFromStream:from: - write:toStream:from: - writeAll:toStream:from:
View Management	- customSaveView - customOpenView
Error Handling	- errorState - errorMessage
Parameter Setting and Retrieval	- setCustomParameter:withValue: - getCustomParameter:
Informational Querries	- getFormatName - copyrightNotice - protocolVersion

INSTANCE METHODS

copyrightNotice:

- (const char *)**copyrightNotice**

This method returns a pointer to a NULL terminated string. This string can be quite long and is used

to report copyright notices, author, where to report bugs, etc¼ It is expected that should a caller wish to use this string into the future, it will make a copy of it. It is not the converters responsibilty to make sure a copy remains around after it has been unlinked.

customOpenView:

- **customOpenView:** (int)*width*

Returns a custom view. This view should not be wider than *width*. This is used so that a converter can request information about the picture it will be asked to open. In general, converters will not need to use this mehod and should return **nil**. A case where this method should be used is when a raw bitmap. In this case, the converter would need to ask information such as width, height, pixel depth, etc¼

See also: - **customSaveView:**

customSaveView:

- **customSaveView:** (int)*width*

Returns a custom view. This view should not be wider than *width*. This is used to request information about the picture about to be saved. If the converter does not need addional information from the user, it should return **nil**. This view will be displayed at the bottom of the save panel. A common use for this method would be to ask about compression method or factor.

See also: - **customOpenView**

errorMessage

- (int)**errorMessage**

Returns an integer describing the current error state of the converter. These can be found at the end of this document.

errorState

- (int)**errorState**

Returns and integer describing the current error state of the converter. This is either none, warning, or fatal. The defines for this can be found at the end of this document.

free

- **free**

Frees up memory used by the converter. This will be called just before un-linking.

getCustomParameter:

- (void *)**getCustomParameter:** (const char *)*parameter*

This returns a pointer to a custom value inside the converter. The use of this method is not highly recommended, but may be necessary. The caller will specify what parameter it wants by send the NULL terminated string *parameter*. The converter should then return a pointer to this value. The caller should make sure to make a copy of this value, as it is not guaranteed to remain stable to extant.

The converter should make sure to always respond to this message. If it does not recognize *parameter*, then it should return **NULL**.

See Also: - **setCustomParameter:forValue:**

getFormatName

- (const char *)**getFormatName**

Returns a **NULL** terminated string given a brief description of the format. This brief description usually just expands upon an acronym. For example, the TIFF converter returns ªTagged Image File Format (TIFF)º.

init

- **init**

Initializes the converter. Many converters may not actually do anything when this is called, but this needs to be here for those that do. Returns **self**.

protocolVersion

- (const char *)**protocolVersion**

This returns a **NULL** terminated string describing the current protocol version supported by the converter. Current converters should return ª1.0º. If you wish to keep this string value around make sure to copy it, as it's existance in the future is not guaranteed nor likely.

readAllFromStream:from:

- **readAllFromStream:** (NXStream *)*stream*
  from: *sender*

Reads images of the converters type from *stream* and return an **NXImage**. Since many formats do not support multiple images in a stream, this method may return **nil**, meaning it is not supported. It will also return **nil** in the event of an error. When an error occurs, the program should message **errorState** and possibly **errorMessage** to find out what's wrong.

The converter may request an instance of **ImageControl** from it's *sender*. The caller should be prepared to respond to all messages send by the converter. For this reason, it's usually better for the programmer to not call this message directly.

See Also: - **readFromStream:from:**
- **errorMessage**
- **errorState**

readFromStream:from:

- **readFromStream:** (NXStream *)*stream*
  from: *sender*

Returns a **NXBitmapImageRep** of the image read from *stream*. The programmer should always try to support this method, but if it does not, it should return **nil**. Also, on an error condition, it should return **nil**. The caller can then querry **errorState** and **errorMessage** to discover the problem.

The converter may request an instance of **ImageControl** from it's *sender*. The caller should be prepared to respond to all messages send by the converter. For this reason, it's usually better for the programmer to not call this message directly.

See Also: - **readAllFromStream:from:**
- **errorMessage**
- **errorState**

setCustomParameter:withValue:

- (BOOL)**setCustomParameter:** (const char *)*parameter*
  withValue: (void *)*ptr*

This sets a custom *parameter* pointed to by *ptr* in the converter. The use of this method is not highly recommended, but here for the support of command line version of the converters. The converter should always attempt to respond to this message. If it is unable to set a value, it should return **NO**, otherwise it should set the value and return **YES**.

See Also: - (const char *)**getCustomParameter:**

write:toStream:from:

- (BOOL)**write:** (id)*image*
  toStream: (NXStream *)*stream*

from: *sender*

This method writes the *image*, represented by an **NXBitmapImageRep** to *stream*. It is expected that this message can send to it's *sender* to request certain data, therefore, the *sender* should be able to respond to all message by the converter. On success, the converter returns **YES** and **NO** on failure.

See Also: - **writeAll:toStream:from:**

writeAll:toStream:from:

- (BOOL)**writeAll:** (id)*images*
  toStream: (NXStream *)*stream*
  from: *sender*

This method writes all *images* out to *stream*. It is expected that this message can send to it's *sender* to request certain data, therefore, the *sender* should be able to respond to all message by the converter. If the converter does not wish to or cannot save multiple images, it should return **NO**. However, it's recommended that at least the first image in the **NXImage** be written to disk, even if this means loss of the other images. A warning message might be sent to inform the user of the loss. This should return **YES** on success and **NO** on failure.

METHODS IMPLEMENTED BY SENDER

getImageControl:

- **getImageControl:** (id)*image*

This method return a new instance of **ImageControl** associated with the **NXBitmapImageRep** *image*. The sender should always make sure to respond to the message. It should return **nil** if it fails to instantiate an **ImageControl**, however, this will most certainly result in the image not be saved or loaded.

CONSTANTS AND DEFINED TYPES

```
/* Converter Error States */
#define CONVERT_ERR_NONE            0
#define CONVERT_ERR_WARNING         1
#define CONVERT_ERR_FATAL           2

/* Converter Error Codes */
#define ERROR_NO_ERROR                                          0
#define ERROR_UNABLE_TO_OPEN        1
#define ERROR_PERMISSION_DENIED     2
#define ERROR_BAD_FORMAT            3
#define ERROR_TRUNCATED_FILE        4
```