Table of Contents

ReadMe

This readme contains basic information about **GrapeCity Documents for Imaging**, also referred to as **GcImaging**.

Release Notes

Refer to the following release notes for the major releases of the product.

For details about latest hotfixes, see the nuget page.

Version 2.2.0.310

Breaking Changes

This version of the product has the following breaking changes.

- Use the GcBitmap.EnsureRendererCreated() method instead of GcBitmap.Renderer property to make sure a non-null instance of BitmapRenderer is returned.
- Renamed GcBitmap.AsBilevelBitmap() method to ToBilevelBitmap (the transparencyMask parameter replaced with the colorChannel parameter, blackIsZero replaced with whiteIsZero with opposite meaning and default value).
- Renamed GcBitmap.AsGrayscaleBitmap() method to ToGrayscaleBitmap() method (the transparencyMask parameter has been replaced with the colorChannel parameter, blackIsZero parameter has been replaced with whiteIsZero with opposite meaning and default value).
- Replaced BlackIsZero property in the BilevelBitmap and GrayscaleBitmap classes with WhiteIsZero property having opposite meaning.
- Replaced blackIsZero parameter of BilevelBitmap and GrayscaleBitmap constructors with whiteIsZero parameter having opposite meaning and default value.
- Removed Image.ConvertToGrayscale() method (instead, you can use the Image.ToGcBitmap() and apply GrayscaleEffect to GcBitmap).
- Moved the Disposed property from the Image class to the IImage interface.
- Replaced Image.AsGcBitmap() method with the IImage.ToGcBitmap() method (IImage interface is supported in Image and various bitmap classes).
- Removed ToPngStream(), ToJpegStream(), GifStream(), FromGcBitmap(), FromFileDeferred(), FromStreamDeferred() and FromBytesDeferred() methods from the Image class. Instead of the removed methods, you can call the Image.ToGcBitmap() and then call any of the GcBitmap.SaveAs() methods to accomplish similar tasks.
- Removed withICC argument from FromFile(), FromStream() and FromBytes() methods of the Image class.
- Added frameIndex as second parameter to the constructors of GcBitmap class which accepts path, stream, or byte array as the first argument.

- Replaced ImageRect type with System.Drawing.Rectangle in method arguments of the GcWicBitmap class.
- Renamed TiffFrame.ReadAsGcBitmap() method to TiffFrame.ToGcBitmap().
- Renamed WicTiffFrame.ReadAsGcWicBitmap() method to WicTiffFrame.ToGcWicBitmap().
- Removed WicImage class (instead, use GcWicBitmap class).
- Removed the Clone() method from the Image class.
- Added the lowerBitsFirst parameter to the Indexed4bppBitmap class constructor.

Changes From the Previous Release

This version of the product has the following changes.

- Image class is now lightweight and contains just the image metadata and a binding to the actual image data (e.g. to a disk file or to a stream).
- Now, users can convert TiffFrame and WicTiffFrame to an Image object.
- GcBitmap and GcWicBitmap can be created from an Image object.
- The IImage interface has been implemented in the following classes: GcBitmap, GcWicBitmap, BilevelBitmap, GrayscaleBitmap, Indexed4bppBitmap, Indexed8bppBitmap.
- Optimized the GcTiffReader and GcWicTiffReader for a scenario wherein users want to load a single frame from a large TIFF file.

New Features and Improvements

The following features have been added with this version of the product.

- Added GcGraphics.DrawRoundRect(RectangleF bounds, Pen left, Pen top, Pen right, Pen bottom, CornerRadius cornerRadius) method, which allows rendering of multi-style rounded border.
- Added GcBitmap.CompositeAndBlend() method supporting all Porter Duff compositing operators and the advanced blending modes for combining two bitmaps into a single image.
- Added AutoLevel(), AdjustLevels(), ExportColorChannel() and ImportColorChannel() methods to the GcBitmap class.
- Added AutoContrast() and AdjustLevels() methods to the GrayscaleBitmap class.
- Implemented the IImage interface in the GcBitmap, GcWicBitmap, BilevelBitmap, GrayscaleBitmap, Indexed4bppBitmap and Indexed8bppBitmap classes.
- TiffFrame and WicTiffFrame can be converted to an Image object.
- Optimized GcTiffReader and GcWicTiffReader for a situation where only frame is loaded from large TIFF file.
- It is possible to load second, third, and other frames from a TIFF file or stream with GcBitmap, GcWicBitmap, and Image classes.
- GcBitmap and GcWicBitmap can be created from an Image object.
- Added constructors to GcWicBitmap class which accepts path, stream, or byte array.
- Added the ToGcBitmap() method overload that accepts an existing instance of GcBitmap to the following classes: TiffFrame, BilevelBitmap, GrayscaleBitmap, Indexed4bppBitmap and Indexed8bppBitmap.
- Added GcBitmap.ToIndexed4bppBitmap() and GcBitmap.ToIndexed4bppBitmap() method overloads that accept a custom palette and a dithering method.
- Added GcBitmap.ToIndexed4bppBitmap() and GcBitmap.ToIndexed4bppBitmap() method overloads based on the Octree quantizer algorithm.
- Added GcBitmap.GenerateOctreePalette() method which creates an Octree quantizer based palette for the current image.
- Added LowerBitsFirst property to the Indexed4bppBitmap class.
- Added Clip() method to the Indexed4bppBitmap, Indexed8bppBitmap, BilevelBitmap, and GrayscaleBitmap classes.
- Added the GcGifReader and GcGifWriter classes that allow users to read and write multi-frame GIF files.
- Added IccProfileData property to the GcBitmap class and other bitmap classes.
- ICC profile can now be loaded and saved to the following formats: JPEG, PNG, TIFF, and GIF.
- Added new constructors to the GcBitmap and GrayscaleBitmap classes that accept existing pixel data to be modified in-place.

- Added ToPngStream() method to the IImage interface and all its related classes.
- Image class is now lightweight. It contains the image metadata and binding to actual image data.
- Added support for all MS Excel pattern fills in the HatchStyle enumeration.
- It is now possible to load any frame (not only the first one) from a TIFF file or stream into GcBitmap, GcWicBitmap and Image objects.

Bug Fixes

The following issue has been resolved since the last release.

- While storing the global palette with less than 129 colors, GcGifWriter doesn't throw any errors now.

Installation

Setting up an application

GcImaging references are available through NuGet, a Visual Studio extension that adds the required libraries and references to your project automatically. To work with GcImaging, you need to have following references in your application:

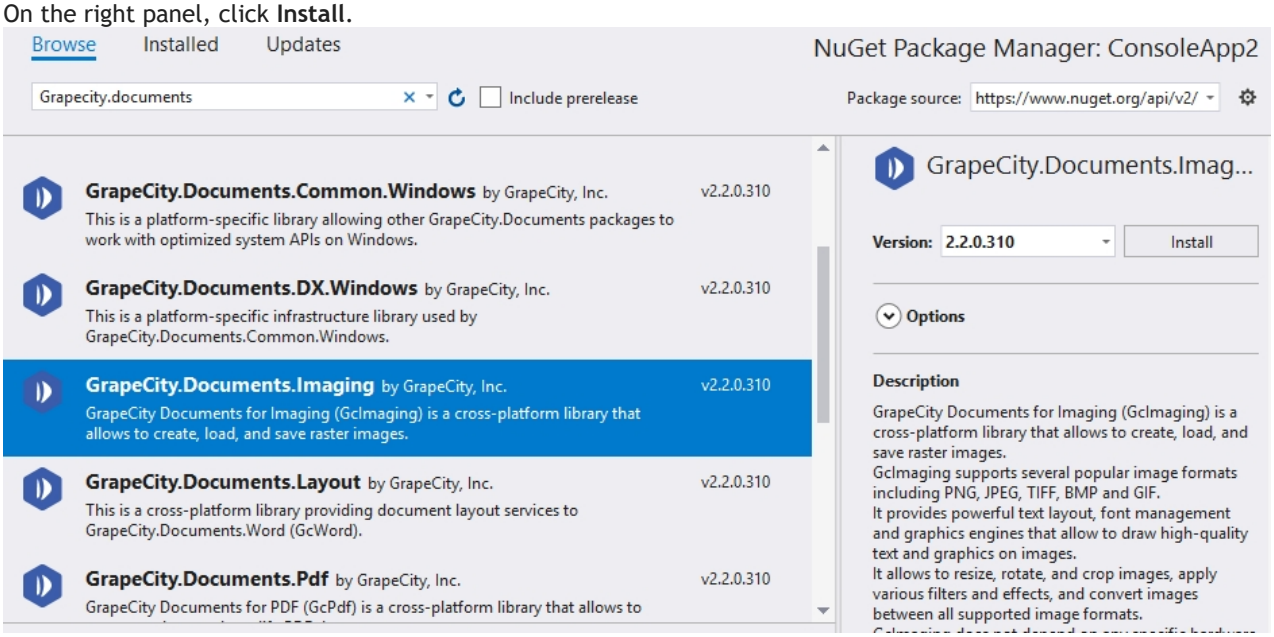
Reference	Purpose
GrapeCity.Documents.Imaging	To use GcImaging in an application, you need to reference (install) just the GrapeCity.Documents.Imaging package. It will pull in the required infrastructure packages.
GrapeCity.Documents.Common	GrapeCity.Documents.Common is an infrastructure package used by GcImaging. You do not need to reference it directly.
GrapeCity.Documents.Common.Windows	On a Windows system, you can optionally install GrapeCity.Documents.Common.Windows. It provides support for font linking specified in the Windows registry, and access to native Windows imaging APIs, improving performance and adding some features (e.g. reading TIFF-JPEG frames).
GrapeCity.Documents.DX.Windows	GrapeCity.Documents.DX.Windows is an infrastructure package used by GrapeCity.Documents.Common.Windows. You do not need to reference it directly.

Add reference to GcImaging in your application from NuGet.org

In order to use GcImaging in a .NET Core, ASP.NET Core, .NET Framework application (any target that supports .NET Standard 2.0), install the NuGet packages in your application using the following steps:

Visual Studio for Windows

1. Open Visual Studio.
2. Create any application (any target that supports .NET Standard 2.0).
3. Right-click the project in Solution Explorer and choose **Manage NuGet Packages**.
4. In the **Package source** on top right, select **nuget.org**.
5. Click **Browse** tab on top left and search for "Grapecity.Documents".
6. On the left panel, select **GrapeCity.Documents.Imaging**
7. On the right panel, click **Install**.



8. In the **Preview Changes** dialog, click **OK** and choose **I Accept** in the next screen.

This adds all the required references of the package to your application. After this step, follow the steps in the Quick Start section.

Visual Studio for Mac

1. Open Visual Studio for Mac.
2. Create any application (any target that supports .NET Standard 2.0).
3. In tree view on the left, right-click **Dependencies** and choose **Add Packages**.
4. In the Search panel, type "GrapeCity.Documents".
5. From the list of packages displayed in the left panel, select **GrapeCity.Documents.Imaging** and click **Add Packages**.
6. Click **Accept**.

This automatically adds references of the package and its dependencies to your application. After this step, follow the steps in the Quick Start section.

Visual Studio Code for Linux

1. Open Visual Studio Code.
2. Install **Nuget Package Manager** from **Extensions**.
3. Create a folder "MyApp" in your **Home** folder.
4. In the Terminal in Visual Studio Code, type "`cd MyApp`"
5. Type command"`dotnet new console`"
   Observe: This creates a .NETCore application with MyApp.csproj file and Program.cs.
6. Press**Ctrl+P**.A command line opens at the top.
7. Type command: "`>`"
   Observe: "**Nuget Package Manager: Add Package**" option appears.
8. Click the above option.
9. Type "**Grapecity**" and press Enter.
   Observe: GrapeCity packages get displayed in the dropdown.
10. Choose **GrapeCity.Documents.Imaging**.
11. Type following command in the Terminal window: "`dotnet restore`"

This adds references of the package to your application. After this step, follow the steps in the Quick Start section.

Product Requirements

GcImaging system requirements, depending upon the framework you are using to create an application, are:

- Our packages include two targets, .NET Standard 2.0 and .NET Framework 4.6.1. In order to use them, your application needs to target either of the following:
  - .NET Core 2.0 or later
  - .NET Framework 4.6.1 or later
- Visual Studio 2015+/Visual Studio for MAC/Visual Studio Code for Linux

For OS versions supported in .NET Core 2.0+, see .NET Core 2.0+ - Supported OS versions.

License Information

GrapeCity Documents for Imaging supports the following types of license:

- **Unlicensed**
- **Evaluation License**
- **Licensed**

Unlicensed

After downloading the product, the product works in the unlicensed mode. However, not more than 10 instances of GcBitmapGraphics and GcWicBitmapGraphics (combined) can be created when the product is used without license.

If you have already created 10 instances of GcBitmapGraphics (BitmapRenderer) and GcWicBitmapGraphics (RenderTarget), following exception is thrown on creating the next instance:

'Unlicensed copy of GrapeCity Documents for Imaging. The number of GcBitmapGraphics (BitmapRenderer) and GcWicBitmapGraphics (RenderTarget) instances is limited to 10. Contact us.sales@grapecity.com to get your 30-day evaluation key.'

Evaluation License

GcImaging evaluation license is available to users for 30 days to evaluate the product. If you want to evaluate the product, you can ask for evaluation license key by sending an email to us.sales@grapecity.com.

The evaluation version has an expiration date that is determined when an evaluation key is generated. After applying the evaluation license key, you can use the complete product until the license expiry date.

After the expiry date, following exception is thrown:

'This evaluation copy of GrapeCity Documents for Imaging has expired. Contact us.sales@grapecity.com to purchase your license. To continue using GcImaging with limitations, remove the expired evaluation license key.'

Licensed

GcImaging production license is issued at the time of purchase of the product. If you have a production license, you can access all the features of GcImaging without any limitations.

Apply License

To apply evaluation/production license in GcImaging, the long string key needs to be copied to the code in one of the following two ways.

- Pass it as an argument to the GcBitmap's ctor:
  ```
  var bmp = new GcBitmap();
  bmp.ApplyLicenseKey("Key");
  This licenses the instance being created.
  ```

- Call a static method on GcBitmap:
  ```
  GcBitmap.SetLicenseKey("key");
  ```
  This licenses all the instances while the program is running.

GcWicBitmap is licensed like GcBitmap, using the same keys and instance counts. Also, please note that if a GcBitmap is converted to GcWicBitmap or vice versa, the converted object will be licensed if the original was.

Redistribution

In order to distribute the application, make sure you meet the installation criteria specified in the Product Requirements page in this documentation. Further, you also need to have a valid Distribution License to successfully distribute the application.

> GcImaging makes it easy to deploy your application to your local servers or cloud offerings such as Azure.

For more information about Distribution License, contact our Sales department using one of these methods:

World Wide Web site	https://www.grapecity.com/
E-mail	us.sales@grapecity.com
Phone	1.800.858.2739 or 412.681.4343
Fax	(412) 681-4384

Viewing Help Content

For detailed information on Documents for Imaging, please refer online documentation.

Technical Support

If you have a technical question about this product, consult the following source:

- Product forum: https://www.grapecity.com/forums
- Email: us.sales@grapecity.com

Contacting Sales

If you would like to find out more about our products, contact our Sales department using one of these methods:

World Wide Web site	https://www.grapecity.com/
E-mail	us.sales@grapecity.com
Phone	(800) 858-2739 or (412) 681-4343 outside the U.S.A.
Fax	(412) 681-4384

End-User License Agreement

The GrapeCity licensing information, including the GrapeCity end-user license agreement, frequently asked licensing questions, and the GrapeCity licensing model, is available online. For detailed information on licensing, refer GrapeCity Licensing. For GrapeCity end-user license agreement, refer End-User License Agreement For GrapeCity Software.