Customer's Canvas Documentation

Introduction

Custom Product Designer is the solution for adding product customization features to print house websites. Unlike a full-fledged web-2-print, the product is designed to be a component for existing sites which are already integrated with e-commerce.

The product is built for the ASP.NET platform and requires Windows hosting. It can be both a dedicated server and a cloud-based solution. Even though this is the ASP.NET application, it allows for seamless integration with any website on any platform. You don't even need knowledge and experience with ASP.NET. You just install the application on your Windows server using our detailed instructions and then use it as a "black box" via our external URL-based API. So if, for example, you have a PHP website integrated with Magento e-commerce and are looking for an online designer, Custom Product Designer can easily fit your requirements.

These are key features of the product:

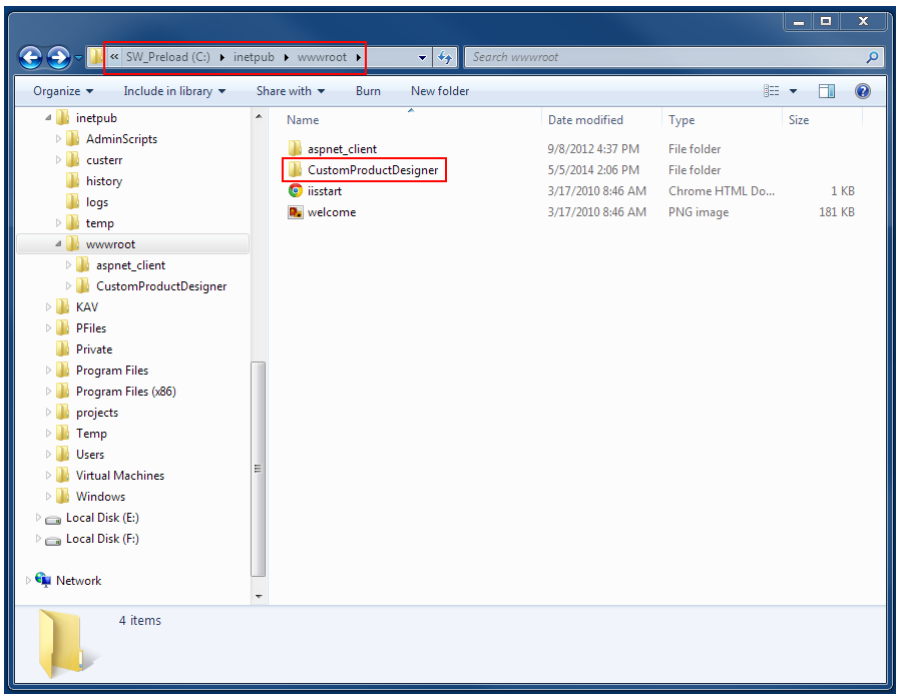
- HTML5 front-end which doesn't require your customers to have Adobe Flash installed on their computers. The solution will even work on mobile tablet Android and iOS devices.
- Adobe Photoshop format for product templates.
- Easy integration with end-users' accounts and orders in your e-commerce.
- Integrated uploader for adding images from end-users' devices to print products.
- Private and public graphics asset libraries for end-users.
- Saving customized products to print-ready PDF files with a specified resolution.
- Support of CMYK products and integrated color management.
- Saving end-user products to the file system and loading them for further editing.
- Easy integration within your workflow.

Deploying application to Windows server

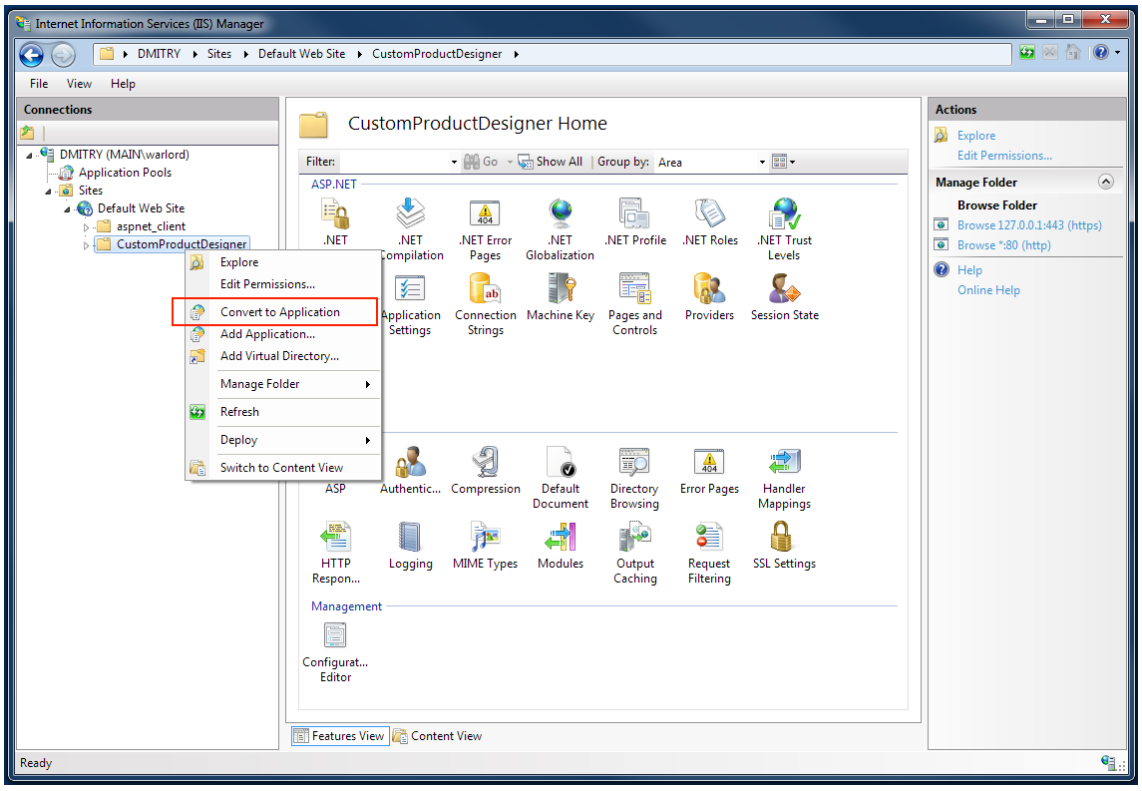
In this chapter we will go through the step-by-step deployment process of Customer Product Designer on the ASP.NET server.

1. First of all, you need to download the application from the customerscanvas.com website. If you don't have a download link, please contact our sales team at sales@aurigma.com

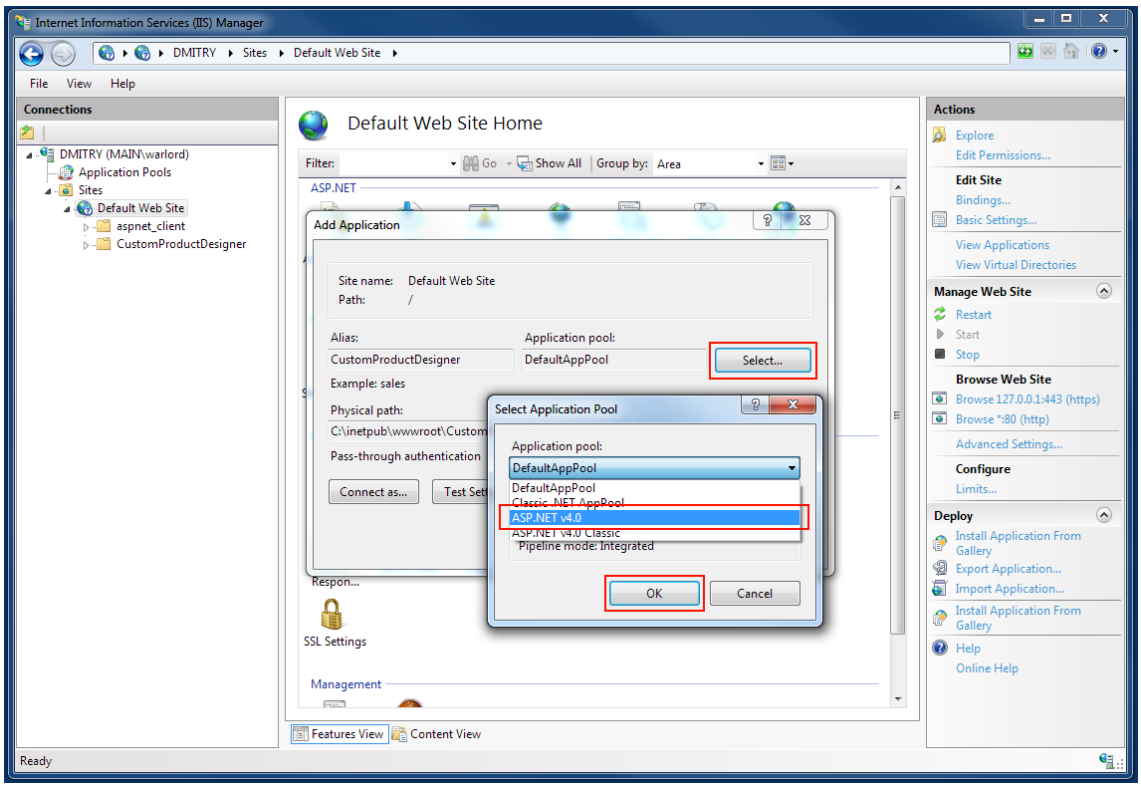
2. After you have downloaded the application you need to unpack it to the /inetpub/wwwroot/ folder.



3. Then you need to register the application within IIS web server. To do that open Internet Information Services (IIS) snap-in and find Custom Product Designer in the left panel. Right click on the application and select the “Convert to Application” menu item.



Don’t forget to choose “ASP.NET v4.0” as an application pool.

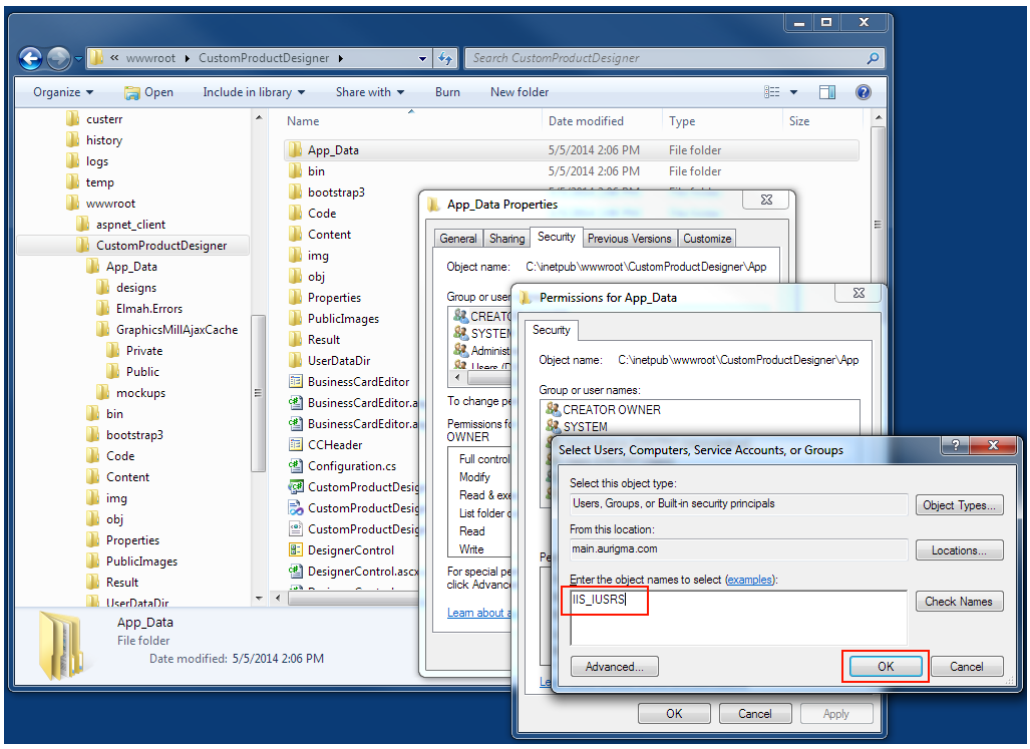


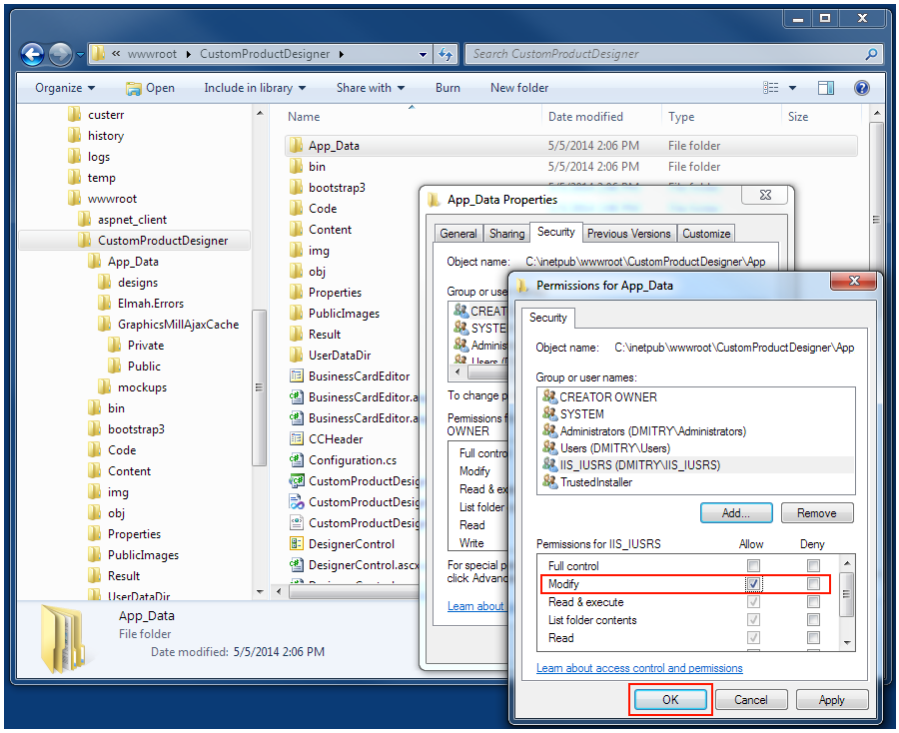
Now you have your application registered within the web server.

4. Now we need to set up file system permissions for the application. The following folders require read/write permission for the account running the web application:

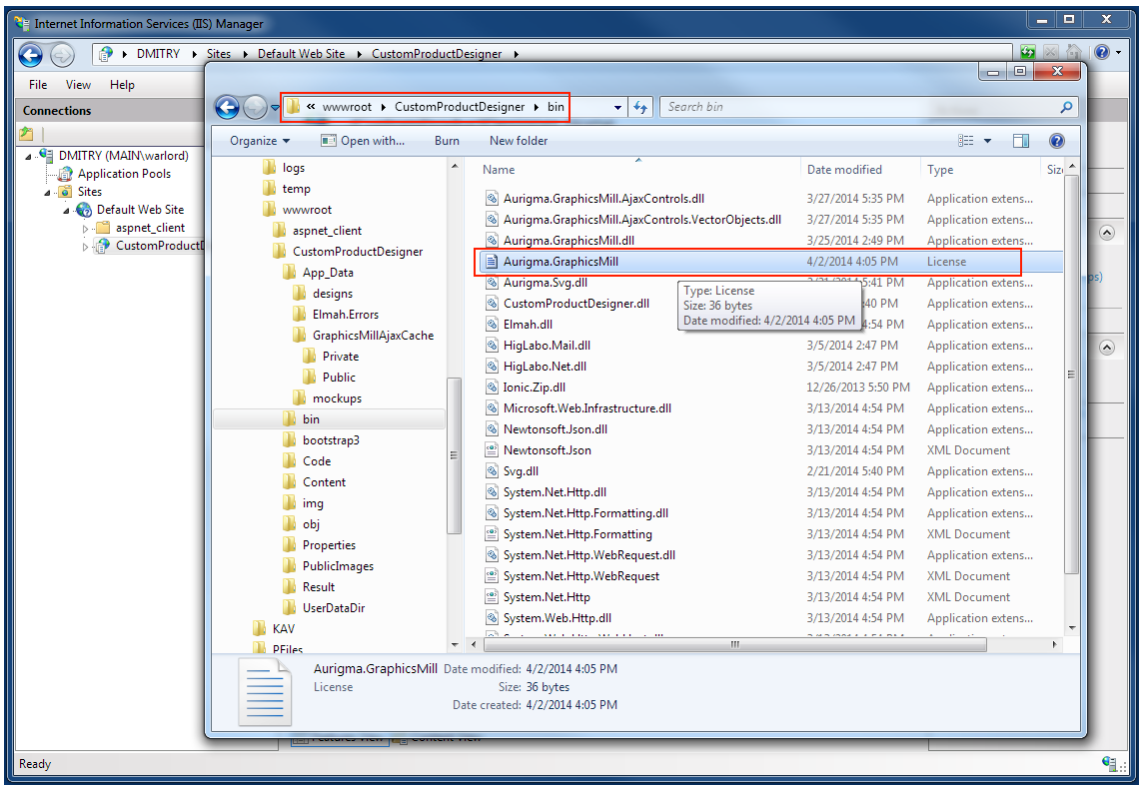
- /App_Data/
- /PublicImages/
- /Result/
- /UserDataDir/

To set up permissions open the file explorer, right click on the folder, and select the “Properties” menu item. Then go to the “Security” tab, click “Edit”, then click “Add”. Type in “IIS_IUSRS”, click “Check Names” and, after it finds the account, press “OK”. Select “Modify” permissions and press “OK” again. Repeat these steps for all the folders on the list.





5. Now we need to register the 30-day trial license key. You should have received Aurigma.GraphicsMill.lic file along with the download link. Put a copy of the file to the /Bin/ folder.



6. After you have done all these steps, open a web browser on your server, type in “http://localhost/customproductdesigner/index.aspx” and you will see the page with links to demo products.

Website integration

As we already know, the designer is hosted as a standalone web application outside of your main website. You embed the designer within a product page on your website using <iframe>. Products and the designer itself are configured through URL parameters. In the next sections we will get familiar with all supported parameters and learn how to create print products. URL query strings can be created dynamically in your application when rendering a product page which may require some programming skills in the platform your main website is built for.

Since the designer is made as a standalone web application, all its resources should be stored on the web server it is hosted on. You may need to transfer files between your main website and the designer, this can be especially important for print-ready files. It can be done in various ways, for example, you may host both of the applications on the same physical server with a shared files system, the servers can have a common shared folder for exchanging data, or your main website can download files from the designer upon completion of the design process.

Configuring Print Products

First off, let’s start with what a print product is comprised of:

- product design,
- product mockup,
- print area,
- print design location.

Product design is an actual print product which is customized by the end-user in their browser. Once the product is completed to the user's satisfaction, the solution renders it to a print-ready file.

Product mockup is a product model which is displayed for demonstration purposes and is excluded from the print-ready file. For example, it can be a t-shirt picture in the t-shirt designer.

Print area is a part of the product which is supposed to be rendered to the print-ready file. Sometimes mockups can be bigger than actual products and this parameter is used for cropping print-ready images to the size of printed graphics.

Design location is usually used along with the previous parameter and defines the point on the mockup the design is displayed.

Only product design is a mandatory parameter, the other parameters can be omitted, especially in case of the product not having a mockup. This would be expected for business cards, greeting cards, etc.

Here is a picture of the bottle label designer with highlighted areas:

Setting up products via query string

As we learned before, products are set up using query string parameters. In this section we will delve into specific parameters and construct a sample product.

The following product-related parameters are supported by Custom Product Designer:

- **Design** – a product design for the 1st product page in Adobe Photoshop format (we will discuss how to create designs in Photoshop in the next section). All product design files are stored in the "\App_Data\Designs\" folder. The product file name should be passed as the argument value without a file extension. For example, if we have the design "\App_Data\Designs\businesscard1.psd", the query string will contain the following parameter "design= businesscard1".
- **Design_page2** – a product design for the 2nd product page. If a value for this parameter is passed in the query string, the editor will enable a special button for switching between product pages.
- **Mockup** – a product model displayed in the designer. Mockups are displayed at design time only and excluded from print-ready files rendered by the software. Mockups are supported by single-

page products only. All mockup files are stored in the "\App_Data\Mockups\" folder and their names are passed as argument values without file extensions. For example, if we have a mockup named "\App_Data\Mockups\tshirt.jpg", we just add the following argument to the query string: "mockup=tshirt". Custom Product Designer accepts JPEG, PNG, and PSD files for mockups.

- **DesignLocation** – indicates the coordinates of the left top corner of the design on the mockup. This argument allows you to precisely position the product design on the mockup in the designer. The syntax for this argument is pretty simple, for example, point (60, 170) will be specified in the following way: "designLocation={X:60,Y:170}".
- **ForegroundMockupEnabled** – this argument allows displaying the mockup above the design. It may be needed when the mockup has transparent areas, e.g. picture frame. Here is how to use this argument: "foregroundMockupEnabled=true".
- **PrintArea** – rectangular area on the product which is rendered to the print-ready file. This argument is used when you have a print product with mockup and only a part of the product is supposed to be printed. For example, it can be used for the t-shirt designer displaying a whole t-shirt picture with the end-user designs graphics on the chest. The syntax for this argument is simple: "printArea={X:63,Y:170,Width:158,Height:288}".
- **VisiblePrintArea** – this argument makes the print area visible in the designer. Here is how to use this argument: "visiblePrintArea=true".

Now let's examine a print product configured with the following URL:

http://.../Editor.aspx?mockup=bottle&design=bottle&designLocation={X:68,Y:175}&printArea={X:68,Y:175,Width:158,Height:288}

As we can see, the question mark is used as a separator between the main part of the URL and arguments. Argument name-value pairs are separated from each other using the ampersand symbol.

The product consists of the mockup and design. They are aligned with each other using the designLocation parameter. The area which goes to the print-ready file is marked with the printArea value.

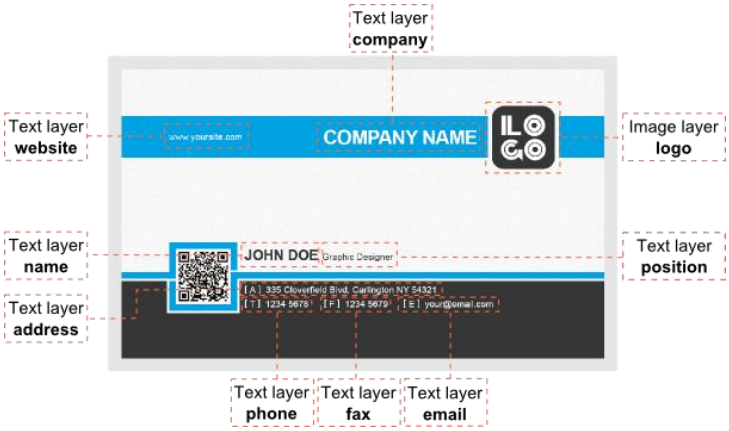
Creating designs in Adobe Photoshop format

Your designers can use the power of Photoshop to create a professional design template and then import it into Custom Product Designer. The user will be able to manipulate its layers, e.g. replace text strings, change positions, etc.

Customer's Canvas cannot load an arbitrary PSD file. The designer should follow the guidelines described below to create a PSD template which can be successfully imported.

Organize layers

Imagine, you are building a template for a business card. You should not put all the text data into a single layer. Instead, the company name should be one layer, the person's name - another one, position - the third one, etc. If you have multiline blocks (e.g. address lines), you should split them into several single-line layers.



Mark up placeholders

Most likely you don't want 100% of the layers to be editable. To specify what text layers should be placeholders for the user's data, add the "**txt_**" prefix to the layer's name. If you don't want the user to modify the layer at all, use the **"locked_"** prefix.

If you want to mark an image layer as background, add the **"background_"** prefix to the layer's name. It will display the image object beneath all other layers and will allow the user to replace it with a solid background.

Avoid mixing placeholders and fixed text. For example, instead of one layer with the text "Phone: [insert your phone]" it is better to use two layers: non-editable "Phone:" and editable "[insert your phone]".

Minimize the number of layers when possible

Avoid creating too many layers which may reduce system performance. For example, it is a good idea to merge all static layers with a background.

Tech requirements

Supported font settings

- Font name (use only those fonts which are installed on the server where you deploy Custom Product Designer);
- Font size;
- Font style (bold, italic, underline);
- Color.

Supported color formats

Color formats may be:

- RGB;
- CMYK;
- Grayscale.

Both 8 and 16 bit-per-channel are supported.

Don't forget to embed the correct color profile into the design (especially if you are using CMYK).

Keep tech limitations in mind

- Only text and image layers are supported. You should flatten all effects or adjustment layers.
- Don't use extra channels.
- Don't use layer properties - visibility flag, lock flag, blending options, etc., are ignored.
- Don't use a layer's visual effects - you should convert them to image layers.
- Don't use a layer mask - apply it to a layer before importing it to Customer's Canvas. Vector masks should be rasterized first.

Working with user accounts

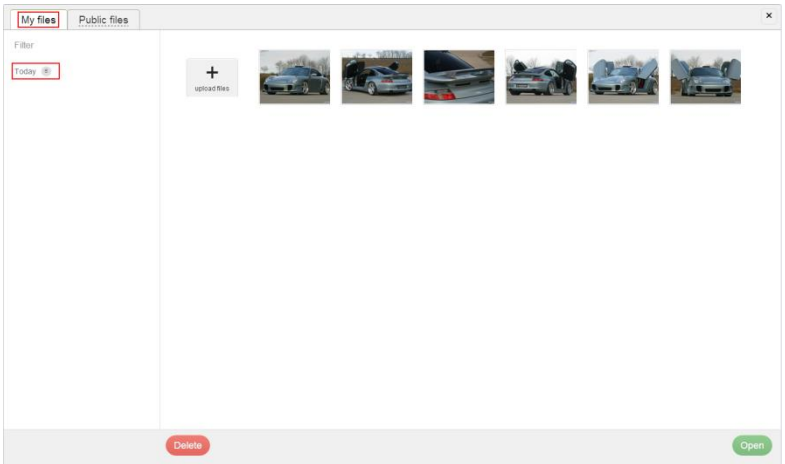
Your website can pass a user ID in a query string to Custom Product Designer:

http://.../Editor.aspx?userID=131415&design=...

The user ID will be used by Custom Product Designer for 1) storing files uploaded by the user and building a private image library, 2) saving a customized product which will allow the user to load them back in the future and continue changing them, and 3) the user ID will be passed back to your website along with the customized product info.

As a value of UserID you can pass any id uniquely referencing users on your website. For example, it can be customer ID in your ecommerce.

Custom Product Designer creates a subfolder in the “\UserDataDir\” for each user ID passed to the system. For example, for the example above we will see “\UserDataDir\131415\” which will have 2 subfolders: “\UserDataDir\131415\images\” and “\UserDataDir\131415\states\”. All image files uploaded by the user will go to the “\images\” and will be available in “My Files” for the user only.



All products created by the user will be saved to the “\states\” subfolder and the user will be able to load them back in the future (we will discuss it later in this topic).

If no user ID passed in the query string, Custom Product Designer uses the user named “default” in this case.

Finishing product customization

Custom Product Designer supports two scenarios for the user to finish customizing the product:

1. The end-user downloads the print-ready file and closes the editor page. This scenario can be easily enabled by the “downloadEnabled=true” argument in the query string.
2. Custom Product Designer submits all the information to your website and your website registers the order in the ecommerce.

In normal operations the second scenario is used for the integration. Here is how it works, after the end-user clicks the “Submit for printing” button, Custom Product Designer does three things:

1. It saves the customized product to the “\UserDataDir\UserID\states\” folder with a unique file name and this file can be reloaded for further editing in the future.
2. It renders the print-ready file and saves it to the “\Result\” folder with a unique file name.
3. Custom Product Designer sends an HTTP POST request to your website containing all the information about the product.

The request goes to the URL configured in <add key="OnCompleteCallbackUrl" value="…"/> in the web.config. The request contains the following JSON formatted string: **{state: [state_file_name], userid: [userid], result: [result_file_name], status: [status]}**, where

- **[state_file_name]** – the file name in the “\UserDataDir\UserID\states\” folder containing customized product.
- **[result_file_name]** – the name of the print-ready file in the “\Result\” folder.
- **[userid]** – the user ID passed in the query string.
- **[status]** – contains the «success» string if the rendering was successful or exception string if there were any errors while finalizing the product.

The server script on your website listening for requests coming to the specified URL should parse the request, and associate the product info with the order in the ecommerce system. Since the name of the print-ready file is passed as a part of the request, your website can either submit the file for downloading or just put the file name in the order, whichever suits you best.

Optionally, you can configure Custom Product Designer to redirect the user to another webpage after the product is completed. The page URL can be set up in the <add key="OnCompleteRedirectUrl" value="…"/> setting in web.config.

As you can see the integration requires the involvement of the programmers who maintain your website. However, if it is needed, the Customer’s Canvas’s technical specialists will be glad to consult with you on any integration related questions, or even implement Custom Product Designer on your website. Feel free to contact us at sales@aurigma.com

Sending email notifications

Custom Product Designer can send email notifications upon finishing the product design. They are enabled with the <add key="EnableSendingEmail" value="True"/> setting in the web.config. Mail server settings are configured in the “\mail.config” file. The file has XML format and contains the following items:

- <SmtpServerName></SmtpServerName> - SMTP server address.
- <Port></Port> - SMTP port.
- <SslEnabled></SslEnabled> - encryption.
- <From></From> - from email address.
- <To></To> - to email address.
- <Login></Login> - SMTP login name.
- <Password></Password> - SMTP password.
- <TemplateFileName>MailTemplate.txt</TemplateFileName> - email message template.

The default template of the email message is stored in the “\MailTemplate.txt” file. The file supports the following placeholders in the message text:

- [USERNAME] – User ID passed in the query string.
- [PDF] – URL to download the print-ready file.
- [STATE] – URL to edit the customized product.

Configuring print-ready files

Custom Product Designer allows configuring the output resolution, color space, and color management settings for print-ready files.

Output resolution is set using the <add key="ProductDpi" value="" /> parameter in the web.config.

The color space of the print-ready output is specified using the “PrintColorSpace” parameter in the query string. It accepts three values: “rgb”, “cmyk”, and “grayscale”. For example, using the following string Custom Product Designer renders CMYK products: “printColorSpace=cmyk”.

Color management

Colors on the design may not match colors on the print-ready output, especially in case you are using the CMYK format. This happens due to various factors, such as the device-dependency of image color formats, inks, paper color, etc., that affect output colors. To negate this effect there is a special technique called color management, and Custom Product Designer supports it.

A system that implements color management must be able to obtain the color characteristics from so called, color profiles. Custom Product Designer requires two types of profiles: source and destination. Destination is the profile for your print-ready output. Source is the profile for any graphics loaded to editor. Different graphics objects may have different profiles. For example, if you load an image to editor, it may have its own embedded color profile which uniquely defines the color characteristics of the pixels.

Custom Product Designer specifies three profiles for grayscale, RGB, and CMYK color spaces in the web.config:

```
<Aurigma.GraphicsMill.AjaxControls.VectorObjects>
  <add key="CmykColorProfileFileName" value="C:\..." />
```

```
    <add key="RgbColorProfileFileName" value="C:\..." />
    <add key="GrayscaleColorProfileFileName" value="C:\..." />
</Aurigma.GraphicsMill.AjaxControls.VectorObjects>
```

By default, profiles specified with these three settings are used as destinations depending on the color space passed in the “PrintColorSpace” parameter. For example, if you render the products to a CMYK print-ready output, the profile configured in CmykColorProfileFileName will be picked as destination one.

As regards source profiles, the rule is simple. If a graphics object has its own embedded profile, it will be used as the source profile. If it does not have an embedded color profile, one of three profiles configured in the web.config will be used as the source profile depending on the color space of the graphics objects. The profile configured in the RGBColorProfileFileName will be picked for RGB objects, CmykColorProfileFileName - for CMYK objects, and GrayscaleColorProfileFileName - for grayscale ones.

If the color profiles are not configured in web.config or configured files are missing, they are set with default values:

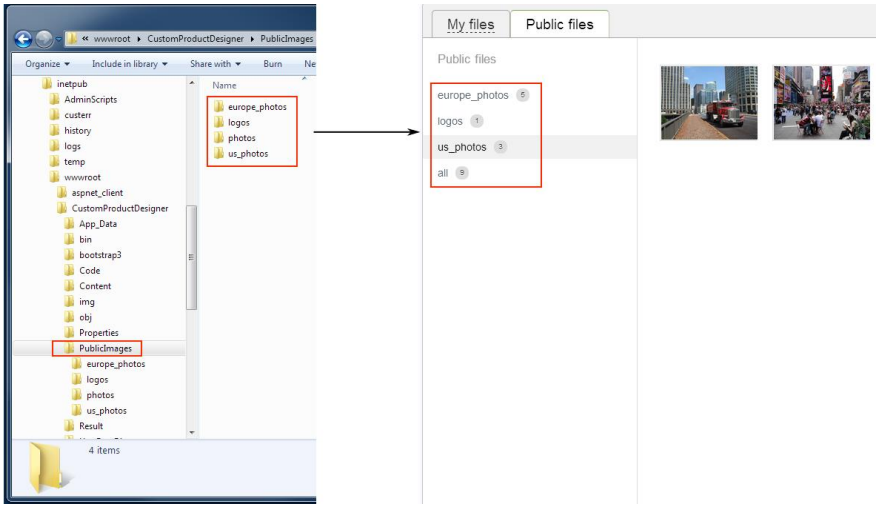
- SWOP (Coated) 20%, GCR, Medium profile for CMYK specified in the Canvas.CmykColorProfile property.
- sRGB IEC61966-2.1 profile for RGB specified in the Canvas.RGBColorProfile property.
- Dot Gain 30% profile for Grayscale specified in the Canvas.GrayscaleColorProfile property.

To enable color management for print-ready output use the <add key="EnablePrintColorManagment" value="True" /> setting in the web.config.

Public image galleries

The solution supports public image gallery which is available for all end-users via the “Public Files” tab in the file picker. All the public image files should be placed in a predefined folder on the webserver where you host Custom Product Designer. By default, public images are stored in the “\PublicImages\” folder, however, you can change their location using the <add key="PublicImagePath" value="PublicImages" /> setting in the web.config. Make sure that the “IIS_IUSRS” user group has read/write access to the folder.

The gallery supports grouping and categorization of the images. You can create subfolders in the folder and organize images between them. It will be reflected in the file picker:



Reloading products for further editing

As we have learned before, upon closing the designer, it saves the product state to a file. The file name is sent in the notification request to the host website in the [state_file_name] field. This way, you can easily store the file name in your system along with other information related to the order. To reload the product in the designer and allow the user to make any changes to it, you just need to pass the UserID and State arguments in the query string: http://.../Editor.aspx?userID=131415&state=05f7be5c-c360-4f7b-8c2a-830ee59fef2a

No other arguments in the query string are needed; all designer settings will be loaded from the state.

Public product templates

As a web-to-print website owner you may want to use the Customer's Canvas designer to create some product templates and then allow customers to customize them. This can be easily done using the "master user" designation. Here is how it works. You design products under the master user with the user ID configured in the <add key="MasterUserId" value="…" /> field in the web.config. You store file names of template states on your website and allow your end-users to open them. They load them using the same technique we considered in the previous section, with only one difference. The URL has the real ID of the current end-user as the UserID value and the file name of the state saved under the master user as the State value. All changes made to the template by the end-user will not affect the original file and you can open the original template for end-users as many times as you want.

Customizing user interface

Here are additional useful query string arguments for customizing the user interface: