ADOBE® scene7®

URL Construction Best Practices

An overview of URL construction for Scene7 image products, updated for the Adobe Scene7 Publishing System and Image Serving/Image Rendering 4.0

Contents

Overview 2

Static/Single Image URLs 3

Viewer URLs 4

Advanced Image Server URLs 6

Vignettes & Render Server URLs 7

Personalization: Nested URLs 11

Technical Documentation 13

Appendix I: Dynamic Image & Render Sets 14

Appendix II: Extending Image Formats with Image Modifiers 21

Appendix III: "Instant" Templates 25

Appendix IV: Zooming on Templates with Flash 30

Appendix V: Zooming on Vignettes with Flash 34

Original Author: Andrew J. Hathaway
Revised and Expanded by
Christopher Fabbro Smith

December 2008
Version 3.0

Overview

This document is intended for Adobe Scene7 Publishing System (SPS) hosted customers, but much of the information contained is relevant to customers that have licensed versions of classic Image Production System (IPS), Image Serving and Image Rendering in-house. This document will present a list of sample URLs as well as some basic explanation of Adobe Scene7 server protocols and syntax.

Terminology
SPS: Scene7 Publishing System
IPS: Image Production System

SPS is the successor to IPS , however both systems interact with the same IPS database, and both publish to the Adobe Scene7 Image Server and Image Rendering Server.

For the purpose of this document, the terms IPS and SPS can be used interchangeably unless otherwise noted.

What's New in this Document?

Added URLs for:

» Zooming on IPS Templates and Vignettes
» Image/Render Sets, calling all IDs in URL
» Image Presets with drop shadow and background color
» Image Presets with layering ("Instant Templates")

Gotchas...

- Adobe Scene7 URL calls are *case sensitive*
- Call only to your assigned server. Your base URL is *not* sample.scene7.com. See sidebars **What's my *real* URL?** and ***Sample* is not your URL!**
- Watch out for typos. It is extremely easy to mistype a command or misspell an image name
- Make sure everything you are calling has been marked for publish, and then run a publish. Not publishing is the #1 cause for broken URLs

Image Servers and the Catalog File

Image Server content is accessed via a URL string. However, unlike most web servers, Adobe Scene7 Image Servers URL calls go to a manifest file rather than to the file proper. The manifest file is a list of all the content published to the Image Server. This list contains a reference to any asset Marked for Publish as well as any metadata associated with it. It contains a path to the image as well.

Understanding this architectural feature, that URL calls are actually made to a text file that references an image rather than to an image directly, allows for the Image Servers to use other metadata stored with the image information.

Sample catalog file

Id	Path	Resolution	Modifier
0033_132x133	ecat/search demo/optimized/0033_132x133_tif.tif	150	crop=75,75,2625,1575
0033_42x43_D2	ecat/search demo/optimized/0033_42x43_D2_tif.tif	150	crop=75,75,2625,1575
0033_44x45_D2	ecat/search demo/optimized/0033_44x45_D2_tif.tif	150	crop=75,75,2625,1575
0033_46x47_D2	ecat/search demo/optimized/0033_46x47_D2_tif.tif	150	crop=75,75,2625,1575
0033_48x49_D2	ecat/search demo/optimized/0033_48x49_D2_tif.tif	150	crop=75,75,2625,1575
0033_50x51_D2	ecat/search demo/optimized/0033_50x51_D2_tif.tif	150	crop=75,75,2625,1575
0033_80x81_D2	ecat/search demo/optimized/0033_80x81_D2_tif.tif	150	crop=75,75,2625,1575
0033_82x83_D2	ecat/search demo/optimized/0033_82x83_D2_tif.tif	150	crop=75,75,2625,1575
0033_84x85_D2	ecat/search demo/optimized/0033_84x85_D2_tif.tif	150	crop=75,75,2625,1575
0033_86x87_D2	ecat/search demo/optimized/0033_86x87_D2_tif.tif	150	crop=75,75,2625,1575
0033_90x91_D2	ecat/search demo/optimized/0033_90x91_D2_tif.tif	150	crop=75,75,2625,1575
0033_98x99_D2	ecat/search demo/optimized/0033_98x99_D2_tif.tif	150	crop=75,75,2625,1575
0033_BC_FCD2	ecat/search demo/optimized/0033_BCD2_FCD2_tif.tif	150	crop=75,75,2625,1575

What's my *real* URL?

Your Welcome Letter contains your assigned base URL, or contact Tech Support (s7support@adobe.com) if in doubt.

Your production URL is also available in SPS via the Copy URL function. Remember that content must be published in order to see it on a live server.

Customers are assigned different base URLs based on two factors:

1. **Location.** North American customer content is served from the U.S., and international customers from the U.K./Europe.

2. **Load-balancing.** To ensure an optimal experience for all customers, URL addresses are assigned different server numbers (e.g., s7d2, s7d3).

Base URLs

Adobe Scene7 offers two base URLs: a direct origin server path and a cached path. Adobe Scene7's On-Demand customers benefit from a cached URL provided by a Content Distribution Network (CDN). For live production web sites, use the CDN-cached URL.

Cached (CDN)

`http://s7dX.scene7.com` (North American [N.A.] customers)
`http://companyname.scene7.com` (UK/European/International customers)
European customers are assigned an Adobe Scene7 subdomain to use as their production URL, e.g., http://adobe.scene7.com might be used by Adobe Systems UK.

Origin/QA

`http://testvipdX.scene7.com` (N.A.) / `http://test-gX.scene7.com` (Europe)
IMPORTANT: Your base URL will have a different and specific number at the end in place of the ***X*** *(s7d4 vs. s7d2, for example) depending on which was assigned to you by Adobe Scene7. For N.A. customers only this number will be the same for both the CDN URL (s7d5.scene7.com) and the testing URL (testvipd5.scene7.com). It is important to use the correct number as assigned to you. When in doubt, contact Adobe Scene7 Tech Support (s7support@adobe.com). Also see the sidebar,* ***What's my real URL?*** *For training purposes only, we will use* ***http://sample.scene7.com/***.

Pyramid TIFFs and You.

When you call to an image URL, you are telling the Image Server to give you an image generated from the Pyramid TIFF master.

Pyramid TIFFs (or P-TIFFs) are special TIFF files that contain multiple sizes or resolutions. They are generated whenever you upload an image. So instead of publishing out your single resolution master, the system publishes out a multiple resolution P-TIFF master.

A command that says MyPhoto?wid=500&fmt=jpg tells the Image Server to create a 500px wide JPEG derived from the MyPhoto P-TIFF master image.

The origin server path will reflect immediately any changes since you last published to the Image Server. Images called via the cached URLs, by default, will refresh every 10 hours. It is important to note that new images will be will seen on both URLs.

URL Construction

Images served from Adobe Scene7 Image Server come from a URL string. Each image is generated from a Pyramid TIFF master image (see sidebar, **Pyramid TIFFs and You**).

Its syntax looks like:
your base URL / name of image server / account name / image name / image modifiers

`http://sample.scene7.com / is/image / S7train / Backpack_A ? wid=250 & resMode=sharp2`

Essentially, the URL contains a reference to the image and any and all Image Server protocols—instructions that tell the Image Server how to act on the image – and this creates the final resultant image. Individual Image Server protocols are concatenated together with an ampersand (&) to create a string of effects. Each protocol 'argument' contains the name of the Image Server effect and the value of the effect. For example, 'wid=250' resizes the image to 250 pixels wide, while 'resMode=sharp2' tells the Image Server to resample the image using a Lanczos interpolation algorithm (sharpens as it resizes).

Another way to look at an Image Server URL is to notice how it is divided by the question mark (?). Everything to the left of the question mark is a virtual path to a specific image. Everything to the right of the question mark are Image Server modifiers, a series of special instructions to the image server telling the image server how to act on the image. This further simplifies the previous construct:
`your path ? image server modifiers`

Static/Single Image URLs

Sample is not your URL!

The URL *sample.scene7.com* is for training purposes only.

Use your assigned production URL given in the Scene7 Welcome Letter.

It will typically be in the format, e.g., `http://s7d5.scene7.com`.

This base URL should be at the beginning of **every** Scene7 URL call.

And origin URLs (in the form of testvipd5.scene7.com) are for QA and testing only.

When in doubt, check your Welcome Letter or contact Tech Support (s7support@adobe.com)

`http://sample.scene7.com/is/image/S7train/Backpack_A`

This calls to this image without passing any specific image protocols after the question mark. The image you see (by default a 400x400 image) is generated from a company's default Image Server publish settings. If no protocols are on the end of the URL, the image server will use these default settings to create an image. You change these settings in IPS by going to Publish>Publishing Settings, and in SPS under Setup > Publish Setup > Image Server.

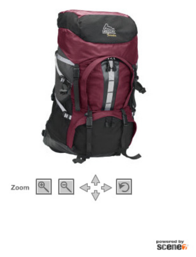
`http://sample.scene7.com/is/image/S7train/Backpack_A?wid=250`
Calls to an image and forces the width to 250 pixels

`http://sample.scene7.com/is/image/S7train/Backpack_A?wid=250&hei=250&fmt=jpeg&qlt=85,0&resMode=sharp2&op_usm=1,1,8,0`
Calls to an image, displays is as a 250x250 pixel wide image as a jpeg with a jpeg quality of 85 (out of 100), sharpens the image while downsampling and then runs an Unsharp Mask filter on the final reply image. Notice that each individual image modifier is concatenated to the next with an ampersand (&) symbol.

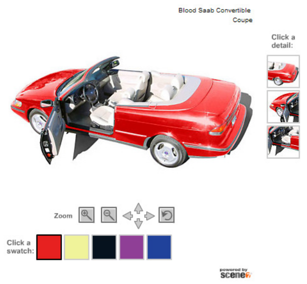
Image Presets (Image Formats)

Image Presets in SPS are the same thing as Image Formats in IPS. The name was changed, but the functionality and usage is the same.

Image Presets allow you to package the various image modifiers most often used to create a dynamically resized image into a small text string. An Image Preset basically contains values for the file format (usually JPEG for the web), pixel count and image sharpening. Instead of appending the URL with each image modifier needed to create a specific type of image size, you create a named Image Preset, such as 'thumbnail', configure the thumbnail Image Preset with the appropriate size, file format and sharpening options and then call to the image using the Image Preset name.



Default zoom viewer



Default zoom viewer with swatches

Basic zoom viewer

DHTML zoom viewer

Spin viewer

These two URLs produce the same 350x350 JPEG image with sharpening:
`http://sample.scene7.com/is/image/S7train/Backpack_A?wid=350&hei=350&fmt=jpeg&qlt=85,0&resMode=sharp&op_usm=0.9,1.0,8,0`

`http://sample.scene7.com/is/image/S7train/Backpack_A?$!_s7product$`

Image Presets can be changed and updated at any time. You will see the results of any change to an Image Preset after you publish again and after the cache for the URL clears. The true value of Image Presets is that if you use one for every image in a size category, any Company Administrator can update the definition of that Image Preset, republish, and affect every image using that format, without changing any web code. **As a best practice, we highly recommend using Image Presets (one Image Preset per unique size on your site).**

Viewer URLs

Zoom Viewers

Adobe Scene7 offers a range of Flash-based and DHTML zoom viewers. In each example below, the URL calls to a JSP page (also hosted on the Adobe Scene7 On-Demand servers) which launches the appropriate viewer and populates it with a specific image or collection of images, such as a paginated eCatalog.

The basic URL syntax looks like this:
your path / name of image server / viewer type **?** account name / sku name

The following is a range of different type of zoom viewers, designed to show a single product image or a set of product images with swatches linked to other product images.

Standard zoom with targets

`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=Backpack_A`

Standard zoom with swatches & targets

`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=color_cars`
(Same viewer as above – flasht_zoom – but calling to a sku that is a Render Set rather than a stand- alone image. The zoom viewer does not care whether you are calling to a set or single image; it can handle either.)

Basic zoom (no swatches/alternate views)

`http://sample.scene7.com/s7/zoom/flashb_zoom.jsp?company=S7train&sku=shoes&zoomwidth=420&zoomheight=470&vc=skin=/is-viewers/flash/basicZoomSkin.swf`
This viewer loads faster, but you must specify a Flash "skin" to be used, as seen in the URL here. It doesn't support swatches or alternate views; it can only display single images.

DHTML zoom (non-Flash based viewer)

`http://sample.scene7.com/s7/zoom/dhtml_zoom.jsp?company=S7train&sku=Backpack_A`
Like basic Zoom, the DHTML viewer also has a small footprint, and now supports zoom targets.

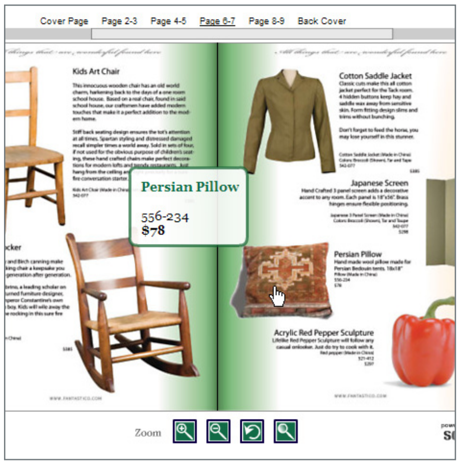
Spin Viewer

The Spin viewer is another type of viewer which calls to a sku which is a series of images designed to be seen from many sides on a turntable.

`http://sample.scene7.com/s7/spin/flash_spin.jsp?company=S7train&sku=cameras`

eCatalog

eCatalogs are Adobe Scene7 products which allow you to repurpose printed material in a familiar book motif. The key feature to this product is the ability to turn "pages" simulating the experience of looking at a multi paged document. This viewer supports zooming as well.



eCatalog using a custom skin and a Formatted Rollover effect (requires interaction with the Adobe Scene7 Info Server)

The URL will look like this:
`http://sample.scene7.com/s7/brochure/flash_brochure.jsp?company=S7train&sku=FantasticoRollover&zoomwidth=790&zoomheight=480`

Similar to other Flash based viewers, you may also create custom configurations in order to offer an alternative look and feel, such as using different Flash based skins, by appending the URL with "config=*name of configuration or eCatalog*."

eVideo

The eVideo viewer serves a Flash Video movie which may be uploaded to SPS and published to an Adobe Scene7 video server. Similar to other viewer URL calls, the eVideo call looks like this:
`http://sample.scene7.com/e2/eVideo.jsp?video=S7learn/Videos/Den-imCo1.flv&height=292&width=340`

Please note that video publishing is a separate event from image publishing, and a video publish must occur before the content can be seen.

Creating Custom Viewer Presets/Configurations

All viewers may be given custom configurations. In addition to changing the behavior of the viewer (e.g., how fast you zoom in/out), you can turn on/off functionality and add sharpening. Here you may also assign a different custom Flash "skin" with a particular viewer so that you may create co-branded looks or offer seasonal changes in the viewers.

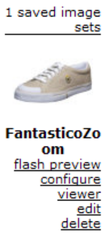
A Flash "skin" is a custom theme that you can apply to the viewer UI. You can create this skin yourself, using Flash, or contract Scene7 to create it for you.

The workflow for configuring the viewers is different whether you are using SPS or IPS. Both methods will create a configuration file which which may be used by any. As a best practice, create a single zoom configuration/preset and call that for every zoom viewer.

If you do not specify a particular configuration to use on the URL calling to a viewer, the Image Server will use your company's default settings. However, by appending to the end of the URL "&config=*configuration name*," the viewer will use your alternate configuration. How you create the configuration (in IPS or SPS), will determine how it is named.

Configuring Viewers in IPS

In IPS, you access the configuration for a viewer by first creating a set (Image Set, Render Set, Spin Set) or eCatalog, and then by clicking "Configure Viewer" on the authoring page.



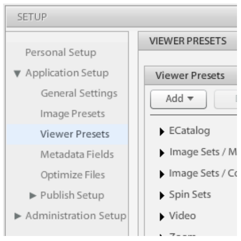
The viewer configuration page in IPS is accessed by clicking the 'configure viewers' link on any set/eCatalog authoring page.

On that page, you choose whether to associate your custom configuration with that viewer, or for all viewers of that type.

When you change the configurations for the first time, you overwrite and update the IPS default configurations for that viewer. This will create a configuration file with the same name as your viewer. For example, if you create an Image Set called "FantasticoZoom," then configure the viewer and save, your config file will also be called "FantasticoZoom."

Configuring Viewers in SPS

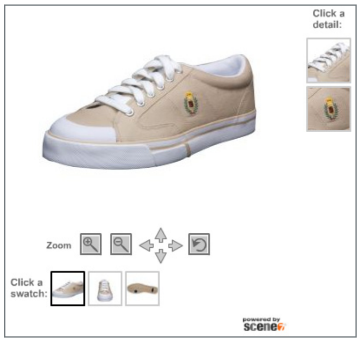
In SPS, all viewer configuration controls are found on the Setup page, under Viewer Presets. It is no longer necessary to first create a set or eCatalog, nor is it possible to use the same IPS method of attaching a configuration to an existing set. Instead, you create a preset which can be used by any viewer in your company. Your configuration file would then be your preset name. For example, if you create a preset "FantasticoZoom," that will be your config name.



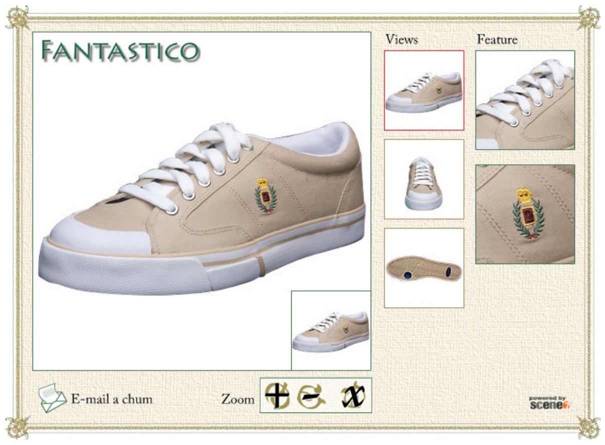
Viewer Presets are accessed under Setup in SPS

Example: Calling an Image Set using default settings
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=FantasticoZoom`

Example: Calling the same Image Set with a custom configuration. The configuration adds the "E-mail a friend" option and calls a custom Flash skin.
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=FantasticoZoom&config=FantasticoZoom&zoomwidth=720&zoomheight=530`



Default configuration (calling default skin)



Custom configuration (calling alternate custom skin)—the Flash skin, sharpening, and options (such as the email feature) are all contained in the `config=` parameter. The configuration is setup in IPS or SPS.

Advanced Image Server URLs

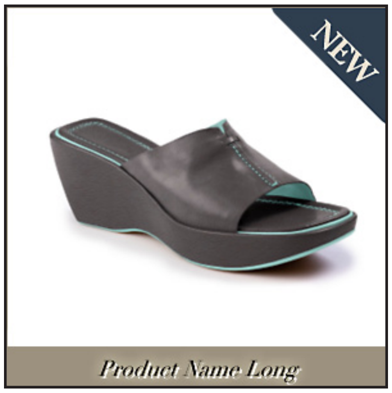
Templates

Templates are layered images which expose various parameters along the URL string. Template layers may contain a mix of images and text layers, and may even contain references to other templates. Using the Template tool in SPS, you may construct a layered file (or edit an imported Photoshop PSD) to allow for any aspect of an image layer or text layer to be addressed. In short, you may change out specific image layers or text attributes by substituting different information. Template parameters are preceded by dollar signs ($).

This is a URL call to the base template:
`http://sample.scene7.com/is/image/S7train/fantastico300`

This URL calls the same template and exposes the parameters. The result is the same image because nothing has changed from the initial values saved with the template:

```
http://sample.scene7.com/is/image/S7train/fantastico300?$!_templ
ate300$&$font=Didot&$red=13&$green=11&$blue=6&$fontsize=34&$text
=Product%20Name%20Long&$pos=104,-104&$sticker=is{S7train/fantas-
tico_new?scl=1}&$product=is{S7train/AP301_CHO_MIN?scl=1}
```



Sample templated image

Changing the parameters in the template will construct an entirely different image.

```
http://sample.scene7.com/is/image/S7train/fantas-
tico300?$!_template300$&$font=Gill%20Sans%20MT&$red=203
&$green=11&$blue=6&$fontsize=34&$text=Canvas%20Sport%20
Shoe&$pos=-104,-104&$sticker=is{S7train/fantastico_
sale?scl=1}&$product=is{S7train/canvas-shoes_3?scl=1}
```

***Sample* is not your URL!**

The URL *sample.scene7.com* is for training purposes only.

Use your assigned production URL given in the Scene7 Welcome Letter.

It will typically be in the format, e.g., `http://s7d5.scene7.com`.

This base URL should be at the beginning of **every** Scene7 URL call.

And origin URLs (in the form of testvipd5.scene7.com) are for QA and testing only.

When in doubt, check your Welcome Letter or contact Tech Support (s7support@adobe.com)

Zooming on Templates
Although the workflow is a little complex, there is an article on zooming on templates after this main document. See Zooming on Templates with Flash (Appendix IV).

Same template calling different parameters

Certain characters are ASCII encoded in the URL string. Notice the space character in between the word 'Canvas' and 'Sport' is encoded in the following part of the URL:
`$text=Canvas%20Sport%20Shoe`

Most web browsers will automatically encode special characters with the ASCII equivalent automatically. However, a plus sign '+' is also an acceptable substitute for a space:
`$text=Canvas+Sport+Shoe`

URL Overrides

The image server will enforce the last image modifier on a URL string if two of the same image modifiers are called. Functionally, you may call an image with an Image Preset and then append the URL with a specific modifier. For example, if you called an image with an Image Preset called $250x250$ (creating a 250 pixel by 250 pixel image), if wid=400 were appended to the URL, the image would be created 250 pixels tall by 400 pixels wide.

An advanced image format can add dynamic layers and drop shadows

Advanced Image Presets

Image Presets (Image Formats) have a property called Image Modifiers, which can extend the functionality of the Image Format. This is explained in detail in *Extending Image Formats with Image Modifiers (Appendix II)* and *"Instant" Templates (Appendix III)*. Here is an example that adds a graphic layer and a drop shadow to a base image:
`http://sample.scene7.com/is/image/S7train/kidsandal2000?$!_50off$`

This feature requires that you have an advanced understanding of Image Serving protocols, because to make it work you will need to manually call out all the needed URL commands. For example the Image Preset above contains this code:

```
&layer=0&size=280,280&extendN=3,3,3,3&effect=-1&op_blur=10&.
effect=Drop Shadow&opac=55&blendmode=mult&pos=5,5&layer=1&src=is{
S7train/50off}&scl=1
```

What happened to Vignette Formats?

In older versions of Image Rendering, the server could not work with pyramid images as the Image Server could. Therefore vignettes had to be published out multiple times for each desired size.

You had to create a "vignette format" for the desired size. You would then call the vignette by calling to the defined format size (example: shoe-300 would give you the 300px size).

This system didn't allow zooming or effective dynamic sizing. The current version of the Render Server converts each master vignette to a pyramid vignette, which contains multiple resolutions. You then call to the desired size.

Vignettes & Render Server URLs

Whereas an Image Server processes bitmap graphic files, a Render Server processes Adobe Scene7 Vignette files. Vignette files are a dynamically renderable file format–an intelligent image– that at the very least allows you to render a prepared file (authored in Adobe Scene7 Image Authoring application) by colorizing certain areas and even applying textures or decals to areas.

Once a vignette file has been authored it is uploaded to IPS and published to a Render Server. On upload, the vignette is converted to a Pyramid Vignette, just as regular images are converted to Pyramid TIFFs. The file stores multiple sizes of itself that can be used for dynamic resizing.

Calling a Vignette

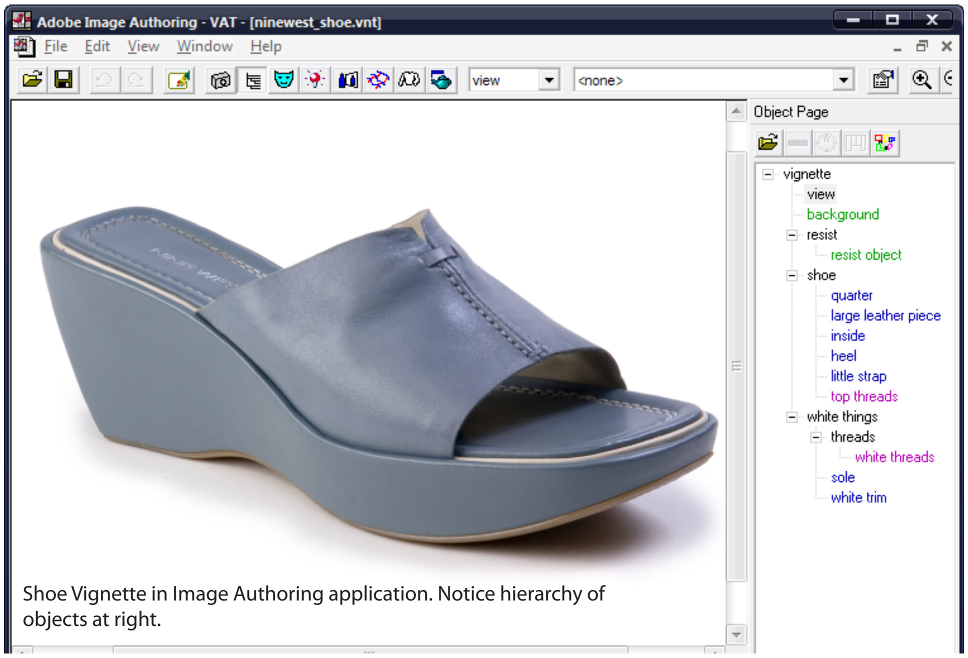
A basic URL calling a vignette will look very similar to one calling to an image except the server type will be `ir/render` (Image Rendering) rather than `is/image` (Image Serving).
`http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=400`

Vignette Naming Convention

To call a vignette, you must understand the composition of that vignette, or else you will get errors. The first step of the authoring process is to build an internal hiearchy of objects. Each group of objects must live in a group (a folder). These objects and groups are typically given useful names that describe their purpose. For example a shirt vignette might be composed of objects called "sleeves," "collar" and "buttons," whereas a car might consist of "doors," "hood" and "antenna." In other workflows, these names might be kept as generic as possible, to make them applicable across a broader range of products: "color1," "color2," or "decal."

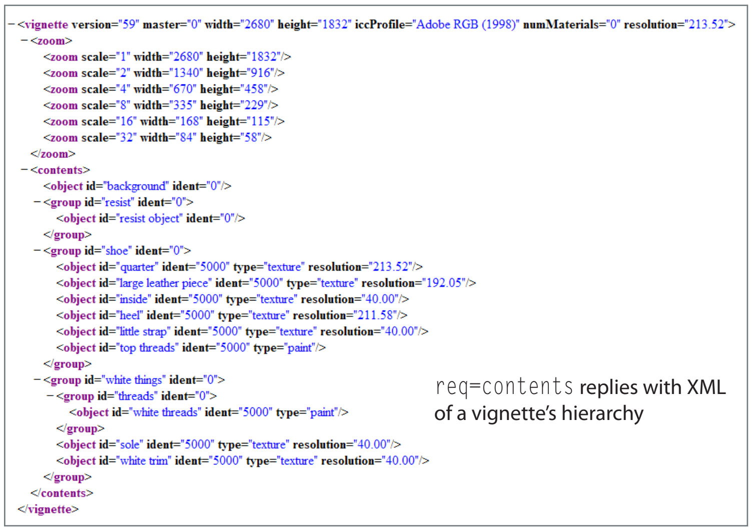
Prior to the authoring work being done, a naming convention should be agreed upon by the designers creating the vignettes and the developers who will be calling to the vignettes via a URL.

However the objects are named will not affect the functionality of the vignette. As a best practice, <u>the naming convention should be established before or during the authoring process</u>, otherwise the graphic artists may come up with arbitrary names for each vignette that must be individually tracked as data by the website team, versus giving them pre-determined names that follow a logical pattern.



Shoe Vignette in Image Authoring application. Notice hierarchy of objects at right.

There is a useful URL call to the render server which will expose a vignette's object hierarchy in XML format. Call to the vignette and append the URL with `req=contents`:
`http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?req=contents`


```
- <vignette version="59" master="0" width="2680" height="1832" iccProfile="Adobe RGB (1998)" numMaterials="0" resolution="213.52">
  - <zoom>
      <zoom scale="1" width="2680" height="1832"/>
      <zoom scale="2" width="1340" height="916"/>
      <zoom scale="4" width="670" height="458"/>
      <zoom scale="8" width="335" height="229"/>
      <zoom scale="16" width="168" height="115"/>
      <zoom scale="32" width="84" height="58"/>
    </zoom>
  - <contents>
      <object id="background" ident="0"/>
    - <group id="resist" ident="0">
        <object id="resist object" ident="0"/>
      </group>
    - <group id="shoe" ident="0">
        <object id="quarter" ident="5000" type="texture" resolution="213.52"/>
        <object id="large leather piece" ident="5000" type="texture" resolution="192.05"/>
        <object id="inside" ident="5000" type="texture" resolution="40.00"/>
        <object id="heel" ident="5000" type="texture" resolution="211.58"/>
        <object id="little strap" ident="5000" type="texture" resolution="40.00"/>
        <object id="top threads" ident="5000" type="paint"/>
      </group>
    - <group id="white things" ident="0">
      - <group id="threads" ident="0">
          <object id="white threads" ident="5000" type="paint"/>
        </group>
        <object id="sole" ident="5000" type="texture" resolution="40.00"/>
        <object id="white trim" ident="5000" type="texture" resolution="40.00"/>
      </group>
    </contents>
</vignette>
```
`req=contents` replies with XML of a vignette's hierarchy

Unlike Image Serving, Image Rendering has no pre-built viewers.

In other words, you cannot use the Image Set, Render Set or Spin Set capability with Rendered Images (vignettes).

It is entirely possible to zoom on a vignette—see Zooming on Vignettes with Flash (Appendix V)*—however you will not be able to group the vignettes together in IPS/SPS and create a set collection.*

Every render implementation has to be custom-built by a web developer, using URL construction. The implementation could be as simple as an HTML form, and as complex as a dynamic Flex or AJAX-based application.

Render URL Construction

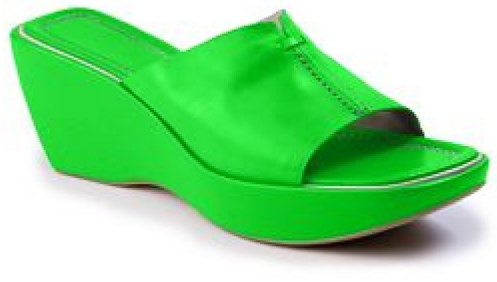
Vignette objects, depending on how they were authored, may be colorized and or mapped with a texture. The texture usually is a repeatable tile, also uploaded and published to a render server. You may build up a render by using the following syntax:

```
object=Object 1&color=RGB value&object=Object 2&src=Texture
```

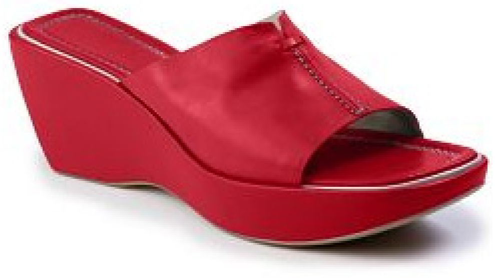
`src=` Calls the IPSID of a bitmapped texture file
`color=` Is an RGB value
`object=` References a vignette group (folder) or object

The exact formatting of the render calls is explained in detail in the online Adobe Scene7 Image Rendering documentation, and is also shown in the following examples.

```
http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=400&obj=shoe&color=40,240,60
```

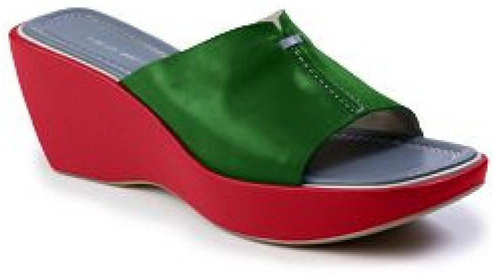


```
http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=300&obj=shoe&color=200,40,60
```



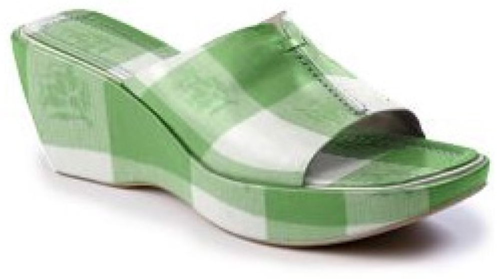
Colorizing the entire "shoe" group, which contains multiple parts

```
http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=300&obj=shoe/quarter&color=200,40,60&obj=shoe/large%20leather%20piece&color=20,60,20
```



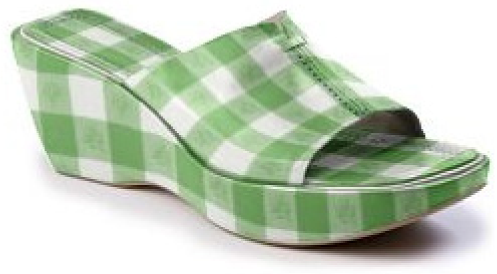
Calling to individual objects in the "shoe" group and colorizing them separately

```
http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=300&obj=shoe&src=185461
```



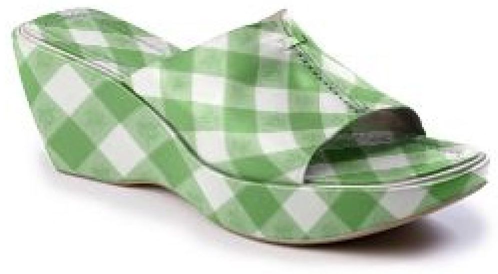
Calling to a texture using the `src=` command

```
http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=300&obj=shoe&src=185461&res=100
```



Resizing the texture as it is mapped on by changing the resolution of the texture (`res=`)

```
http://sample.scene7.com/ir/render/S7trainRender/ninewest_shoe?wid=300&obj=shoe&src=185461&res=100&rotate=45
```



Rotating the texture 45 degrees with the `rotate=` command

Sample is not your URL!

The URL *sample.scene7.com* is for training purposes only.

Use your assigned production URL given in the Scene7 Welcome Letter.

It will typically be in the format, e.g., `http://s7d5.scene7.com`.

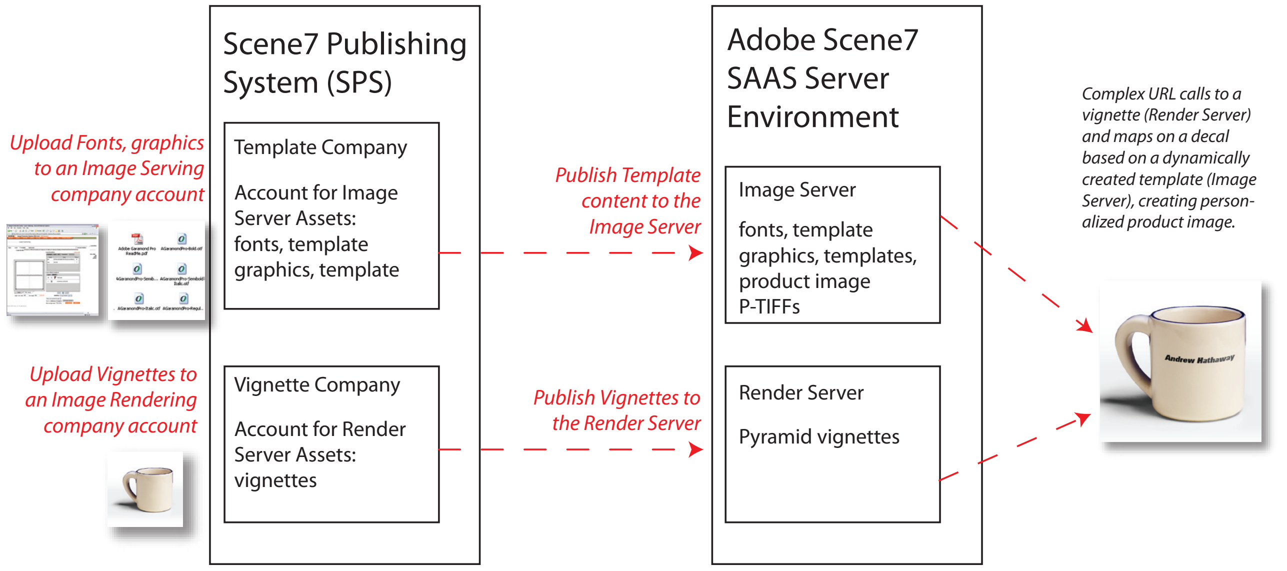
This base URL should be at the beginning of **every** Scene7 URL call.

And origin URLs (in the form of testvipd5.scene7.com) are for QA and testing only.

When in doubt, check your Welcome Letter or contact Tech Support (s7support@adobe.com)

Personalization: Nested URLs

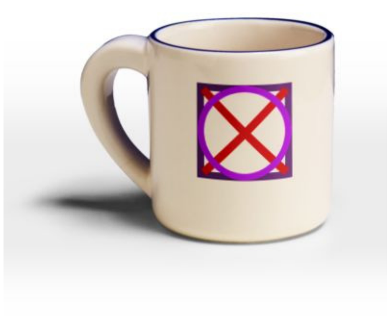
You may work with both types of assets, images and vignettes, and by extension call to both types of servers to create a nested URL. Usually this is done in order to create a personalized product image. The product image in this case is a vignette with an area mapped to take a decal, served from the Render Server. The decal can be a graphic of a paramaterized text layer from a template rendered from the Image Server. When the two URLs are combined, it creates a personalized product image.


Upload Fonts, graphics to an Image Serving company account
Upload Vignettes to an Image Rendering company account
Scene7 Publishing System (SPS)
Template Company Account for Image Server Assets: fonts, template graphics, template
Vignette Company Account for Render Server Assets: vignettes
Publish Template content to the Image Server
Publish Vignettes to the Render Server
Adobe Scene7 SAAS Server Environment
Image Server fonts, template graphics, templates, product image P-TIFFs
Render Server Pyramid vignettes
Complex URL calls to a vignette (Render Server) and maps on a decal based on a dynamically created template (Image Server), creating personalized product image.
Andrew Hathaway

This URL calls to a render server displaying the default image for a vignette of a mug.
`http://sample.scene7.com/ir/render/S7trainRender/Cup1?wid=400`

This URL calls to the vignette and maps on a 'decal', also published to the Render Server.

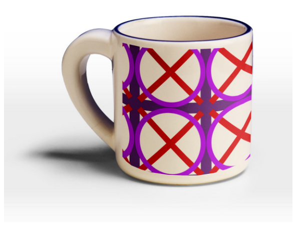
```
http://sample.scene7.com/ir/render/S7trainRender/
Cup1?wid=400&obj=FLOW/MNGRM&decal&src=TESTDECAL&res=100
```



By default, when you apply a bitmap source file the Render Server will tile the image.

When working with personalization, you use the command `&decal` *to tell the Render Server* ***not*** *to repeat the texture.*

If you forget to include this command, you will get something like this:



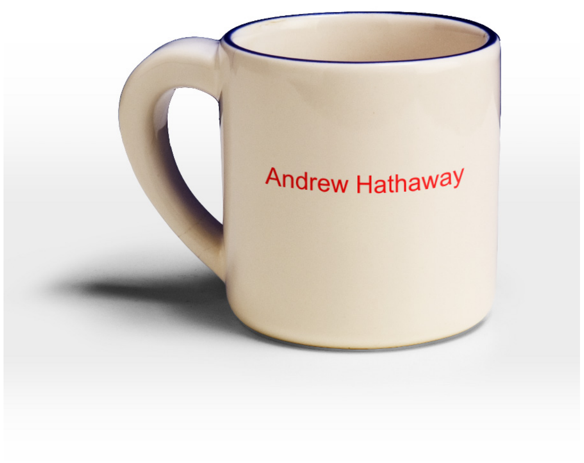
Result of not using the `&decal` command to prevent image tiling

The nested URL syntax for calling a template on a vignette looks like this:

```
Render Server Path / Company name / Vignette Name ? object=name
& decal & source= Image Server { Company Name / Template or Image
Name ? Image Modifiers & Exposed Parameters }
```

This is a nested URL which results in a personalized mug image:

```
http://sample.scene7.com/ir/render/S7trainRender/
Cup1?wid=400&obj=FLOW/MNGRM&decal&src=is{S7train/15w_tmp?scl=1&
$font=Arial&$red=255&$green=10&$blue=10&$size=70&$text=Andrew%20
Hathaway}&res=100
```



And passing different values for the parameters such as color, fonts and even name text results in a different type of render.

```
http://sample.scene7.com/ir/render/S7trainRender/
Cup1?wid=400&obj=FLOW/MNGRM&decal&src=is{S7learn/15w_tmp?scl=1&$f
ont=Radiance&$red=10&$green=10&$blue=255&$size=60&$text=Andrew%20
Hathaway}&res=100
```

Sample is not your URL!

The URL *sample.scene7.com* is for training purposes only.

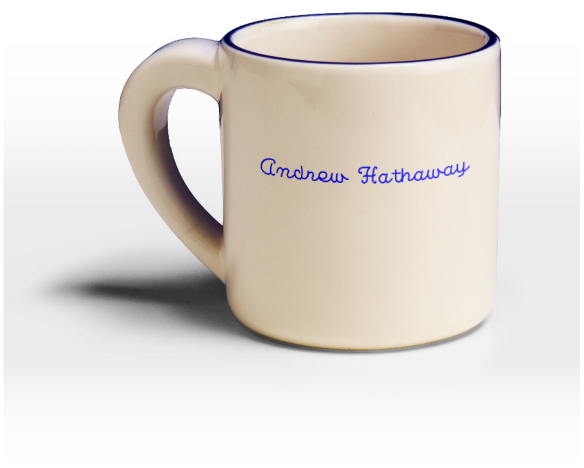
Use your assigned production URL given in the Scene7 Welcome Letter.

It will typically be in the format, e.g., `http://s7d5.scene7.com`.

This base URL should be at the beginning of **every** Scene7 URL call.

And origin URLs (in the form of testvipd5.scene7.com) are for QA and testing only.

When in doubt, check your Welcome Letter or contact Tech Support (s7support@adobe.com)



About this document

All of the example URLs are working URLs, created from various published assets on their appropriate servers. You may copy and paste them in into any web browser and try them for yourself.

SPS Limitations versus Server Options

SPS is a web based application intended to offer a large range of options with a simple to use interface. However SPS does not expose all of the options available to you. Adobe Scene7 Image and Render Servers offer more options than are present in SPS applications. Usually, these are used for advanced imaging projects such as Image Server template creation and personalization projects, which combine aspects of both types of servers.

To learn more about the full range of rendering options for both types of servers, as well as specifications on various viewers, see the online technical documentation, referenced in the next section.

Technical Documentation

Adobe Scene7's technical documentation for both the Image Servers and the Render Servers are available online. These HTML pages detail the range and collection of server protocols. Please access these by using the following URLs.

Image Server

`http://crc.scene7.com/is-docs-4.0/`

Render Server Documentation

`http://crc.scene7.com/ir-docs-4.0/`

Image Server Viewers

`http://crc.scene7.com/is-viewers/`

For general questions contact
***Scene7 Technical Support**:*
s7support@adobe.com

For additional training or for help with advanced topics, contact the
***Scene7 Training Department**:*
scene7training@adobe.com

Appendix I: Dynamic Image & Render Sets

Overview

This is an introduction to creating Image Sets and Render Sets from a URL string, versus manual creation in the Adobe Scene7 Image Production System (IPS).

For the purpose of this document, it is assumed that the reader already understands how sets are manually created in IPS. Basic understanding of Scene7 URL construction and use of viewers is also necessary to understand the concepts presented here. This document builds on knowledge presented in the *URL Construction Best Practices*.

Image and Render Sets are Scene7 image products, typically created in IPS manually or via an automation script, that allow grouping related images into a single viewer. Typically, clients use Image Sets to display alternative views. They are represented by a series of image thumbnails and a main zoom image. By clicking on a thumbnail, the viewer switches to a different image.

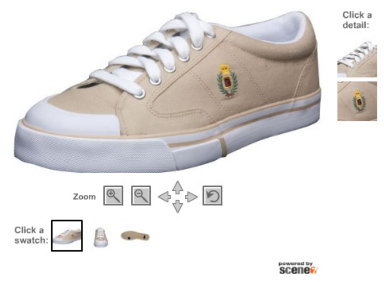
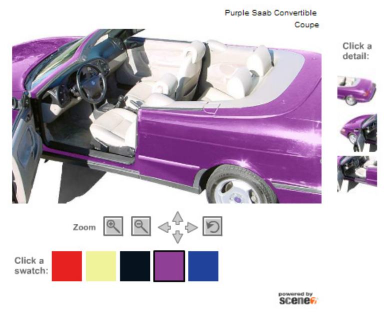


Image Set, used for alternate views

Render Sets are similar to Image Sets, but are typically used as a swatching application. In a Render Set, colored product images are associated with sample color swatches. The user interface for the viewer has a single zoom image and a series of swatches. By clicking on a swatch, the view changes to the product of that color.



Render Set, used for swatching

By default, images in the render set are grouped in twos. To group together larger sets of images, set the Render Set Size setting to a higher number. A user can change this setting to group a larger number of swatches with a single renderable image. Or alternatively, a user can make the Render Set emulate an Image Set by changing the Render Set Size to "1."

However, Image and Render Sets have a distinction regarding the use of zoom targets. Unlike Render Sets, Image Sets do not share zoom targets. If any of the images in the Image Set have zoom targets defined, the user will be able to zoom into them only when that image is selected in the viewer. With Render Sets, zoom targets are automatically associated with all the images in that render set. Therefore, a single zoom target on an image in a Render Set will appear on all images, but in a Render Set that same zoom target would only appear on that particular image.

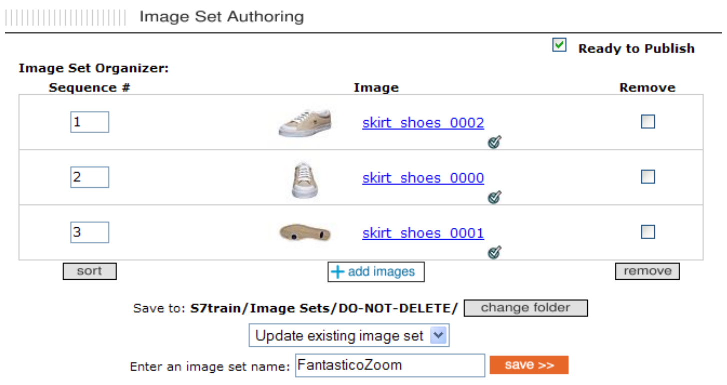


Image Set Authoring

☑ Ready to Publish

Image Set Organizer:

Sequence #	Image	Remove
1	skirt_shoes_0002	☐
2	skirt_shoes_0000	☐
3	skirt_shoes_0001	☐

Save to: S7train/Image Sets/DO-NOT-DELETE/

Update existing image set

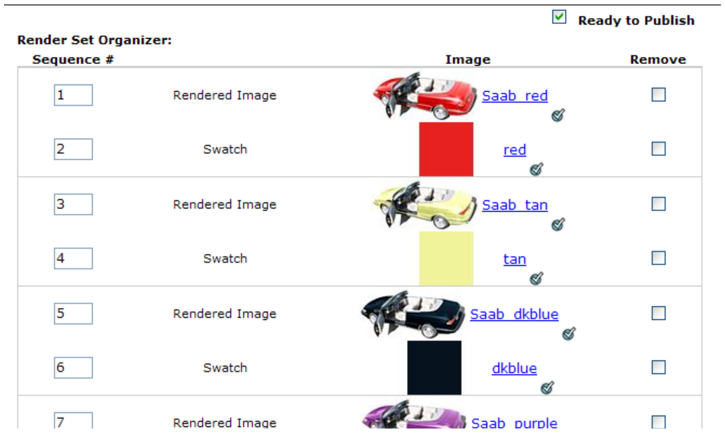
Enter an image set name: FantasticoZoom

Image Set authoring page in IPS

Manual Set Authoring in IPS

Between Image Sets and Render Sets, Image Sets are the least complex to build in IPS. A user selects a group of images, creates a new Image Set, sequences and saves them with a unique name. On the URL, that name is given to the Flash viewer and it returns the set.

Render Sets have an extra authoring step. A user would not only select the images to include (in as many colors as desired), but also the swatch images that will be seen as part of the UI. The user then associates a colored/rendered image with its corresponding swatch (e.g., matching up the blue car with the blue swatch, red car with the red swatch). The user then saves this set under a unique name and that is used as part of the URL.



☑ Ready to Publish

Render Set Organizer:

Sequence #		Image	Remove
1	Rendered Image	Saab_red	☐
2	Swatch	red	☐
3	Rendered Image	Saab_tan	☐
4	Swatch	tan	☐
5	Rendered Image	Saab_dkblue	☐
6	Swatch	dkblue	☐
7	Rendered Image	Saab_purple	☐

Render Set authoring page in IPS

IPS then creates a folder to contain each Image and Render Set, and tracks each as a separate asset. The images themselves are stored under the Images folder (as are all Pyramid TIFF images), but the sets—which are essentially database line-items—are saved in a virtual folder structure. When IPS publishes to an Image Server, it publishes the underlying images plus a reference to the sets in the Image Serving catalog file, with instructions how the sets are constructed. For example, a Render Set consisting of three colored images of a car might have a line item referring to blue_car /blue_swatch, red_car/red_swatch, and green_car/green_swatch.

Pros and Cons of Manual Set Construction

From an implementation perspective, all that the client's database would have to know is the name of the set. It doesn't have to be aware of the name of the individual images. Under this workflow, the end users are responsible for creating the sets in IPS and making sure they are correctly formed before they appear on the website.

This approach works well as long as the content of the sets do not need to change frequently. To make an edit to a set, a user would have to go into IPS, make the change, save and republish. On the Scene7 On-Demand hosted environment, this would then take 10 hours for the change to propagate on the Content Distribution Network (CDN).

Creating Image and Render Sets in IPS means that the website content management system does not need to keep track of the names of the individual images associated with a product.

However if the images need to change frequently within the sets, this method can be limiting.

For some clients, it is not feasible to make manual edits to the sets. Here are some factors that might make this approach unrealistic:

- ***Sheer number of products/SKUs.*** Having to make manual changes in thousands of sets would be incredibly time-consuming.
- ***Dynamic inventory control.*** As customers purchase products on an ecommerce site, the client might want the viewer to hide out-of-stock products.
- ***Decentralized workflow or small production staff.*** The creation of manual sets assumes that there are dedicated users who can manage edits to the sets, but if this resource is not available or the workflow is distributed across multiple parties, this may not be an option.

If a client has a robust content management system/database that can track individual images in a set, or has a solid naming policy that can predictably call related images based on a naming convention, then dynamic set creation may be an option.

How IPS Views Sets

To create sets via the URL, it is necessary to understand how IPS stores information about sets. Here is a line item for a Render Set, "color_cars" from the S7train IPS company catalog, the manifest of file information it sends to the Image Server:

Render Set "color_cars" in S7train account
```
S7train/Saab_red;S7train/red,S7train/Saab_tan;S7train/
tan,S7train/Saab_dkblue;S7train/dkblue,S7train/Saab_
purple;S7train/purple,S7train/Saab_blue;S7train/blue
```

Note how IPS groups the items. It groups the first pair of images by separating them with a semicolon (;):
```
S7train/Saab_red;S7train/red
```

These are the IPSIDs for the image of the red car (Saab_red), and for the red swatch (red). It then separates the pairs with a comma (,):
```
S7train/Saab_red;S7train/red,S7train/Saab_tan;S7train/tan
```

These are the red pair and the tan pair.

So we see that the syntax for a Render Set is
`imageA;swatchA,imageB;swatchB, … imageZ;swatchZ`

Internal Render Set syntax:
imageA;swatchA,
imageB;swatchB,
…
imageZ;swatchZ

Here is the catalog entry for an Image Set, "shoes," in the S7train account:
`S7train/shoe1;S7train/shoe1,S7train/shoefront1;S7train/shoefront1,S7train/shoeside1;S7train/shoeside1`

Here we see the same syntax as above, but notice that each image is called twice:
`S7train/shoe1;S7train/shoe1`

Essentially, the viewer is asking the same image to represent not only the zoom image but the swatch, or thumbnail as well.

Internal Image Set syntax:
imageA,
imageB,
…
imageZ

Pairs of images are separated with commas, as before:
`S7train/shoe1;S7train/shoe1,S7train/shoefront1;S7train/shoefront1`

This represents the 2/3 and front views of the shoe.

So the Image Set syntax is almost identical to the Render Set syntax, but instead of calling swatches, the same image is called twice:
`imageA;imageA, imageB;imageB, … imageZ;imageZ`

As a side note, this method of construction is based on a legacy standard that required a pair of images be sent to the viewer to properly display both a main image and a thumbnail. This is no longer required by the viewers, so for simplicity an Image Set can also be expressed in this way with no loss of functionality:
`imageA, imageB, … imageZ`

For the purpose of this document, it is assumed that the two methods are interchangeable and the simpler method will be used.

Zoom Viewer URLs

There are two methods for calling Scene7 zoom viewers.

What is s7ondemand?
s7ondemand is a viewer platform consisting of pre-configured Java Server Pages (JSP), hosted on the Scene7 On-Demand network.

They simplify calling zoom, spin, and eCatalog products by pre-configuring and embedding the Flash viewer in a web page, and exposing the least number of parameters required by the viewer.

The simplest way to call a standalone image or an existing Image Set or Render Set saved in IPS is to call to the hosted s7ondemand ("pop-up") zoom viewer. For this viewer, all that is needed is the name of the company and the IPSID of the image or set (as a best practice, we recommend that the configuration for that viewer is also called every time).

Here are s7ondemand URLs for the above-mentioned Image and Render Sets:

Render Set (s7ondemand)
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=color_cars&config=color_cars`

Image Set (s7ondemand)
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=shoes&config=shoes`

The second method is to call to the viewer as a Flash object embedded in a web page. This method is more complex because several different parameters must be called on the URL, and a web page must be created to host the viewer. Here are the same URLs, using the embedded syntax:

Render Set (embedded URL)
`http://sample.scene7.com/is-viewers/flash/genericzoom.swf?serverUrl=http://sample.scene7.com/is/image/&contentRoot=http://sample.scene7.com/skins/&image=S7train/color_cars&viewSize=350,350`

Image Set
`http://sample.scene7.com/is-viewers/flash/genericzoom.swf?serverUrl=http://sample.scene7.com/is/`

`image/&contentRoot=http://sample.scene7.com/skins/&image=S7train/shoes&config=S7train/shoes&viewSize=350,350`

The embedded syntax requires more pieces of data, even though the net result is the same. Note the `&image=` command which calls the name of the company account and the name of the Image Set/Render Set. To simplify, the syntax for loading the image or set is:

`&image=companyName/IPSID`

Where *companyName* is the account name in IPS, and *IPSID* is the name of the standalone image, Image Set or Render Set. This command is key to calling dynamic sets, as will be explained shortly.

The s7ondemand JSP uses this same syntax within its own HTML, but only exposes the required parts, like the IPSID (`sku=`) and the company account name (`company=`), and allows optional arguments like the configuration command.

Getting Dynamic Set Information

Did you know you get the contents of a set dynamically, via a URL? All you need to do is call to set as if it were an image, and add the command `req=imageset`.

Here is a sample request:

`http://sample.scene7.com/is/image/S7train/shoes?req=imageset`

And the Image Server replies back with information from the manifest:

`S7train/shoe1;S7train/shoe1,S7train/shoefront1;S7train/shoefront1,S7train/shoeside1;S7train/shoeside1`

Why would you want to do this if you plan to build your sets dynamically? In some workflows, image specialists will build sets in IPS, however these images are then called dynamically so they can be "massaged" before being delivered to the web page (perhaps using Ajax, Flex or some kind of rich media web techology)

Programmatically Creating Sets via the URL (Embedded Method)

Dynamic set creation is generally a little simpler and easier to understand using the embedded syntax than it is with the s7ondemand syntax.

As explained earlier, IPS stores the sets in this general format:

Render Sets
`imageA;swatchA,imageB;swatchB…imageZ;swatchZ`

Image Sets
`imageA,imageB,…imageZ`

And when calling a set in a viewer, the `image=` argument is used:
`&image=companyName/IPSID`

Therefore to load a set programmatically, call to the string of images instead of the single Image/Render Set.

Dynamic Image Set (embedded URL)
`http://sample.scene7.com/is-viewers/flash/genericzoom.swf?serverUrl=http://sample.scene7.com/is/image/&contentRoot=http://sample.scene7.com/skins/&image=S7train/shoe1,S7train/shoefront1,S7train/shoeside1&config=S7train/shoes&viewSize=350,350`

Dynamic Render Set (embedded URL)
`http://sample.scene7.com/is-viewers/flash/genericzoom.swf?serverUrl=http://sample.scene7.com/is/image/&contentRoot=http://sample.scene7.com/skins/&image=S7train/Saab_red;S7train/red,S7train/Saab_tan;S7train/tan,S7train/Saab_dkblue;S7train/dkblue,S7train/Saab_purple;S7train/purple,S7train/Saab_blue;S7train/blue&config=S7train/color_cars&viewSize=350,350`

Note that each call still has a configuration (`config=`) command. This is a best practice, as the configuration typically includes sharpening, any desired viewer options and could include a custom Flash skin as well. Because using this dynamic method means that sets would not be created in IPS, it is recommended that each client set up a single Image or Render Set to use as a default configuration.

For example Training might set up an Image Set called "s7trainzoom," assign it sharpening, a custom skin and turn on the "email a friend" feature, all on the Viewer Configuration page in IPS. Upon publish, every set in the S7train account could call `&config=s7trainzoom`, and know that all those options would be applied without having to create the individual Image/Render Sets.

Also notice that zoom targets in the Render Set example are not working as expected; each image is treating zoom targets separately. If zoom targets are part of the desired implementation, then some extra steps are necessary. This issue takes some explanation and will be addressed later in this document (see below).

Programmatically Creating Sets via the URL (s7ondemand Method)

The embedded zoom viewer call exposes the `image=` command, so it is relatively easy to see where to put this information. When calling viewers using s7ondemand, the process is a little less straight-forward.

The s7ondemand sku= command will not accept multiple images.

The `sku=` command on s7ondemand does not accept multiple images. The following command is ***not*** allowed:
`sku=S7train/shoe1,S7train/shoefront1,S7train/shoeside1`

There is no way to use the `sku=` parameter to send multiple images to the s7ondemand viewer, and the s7ondemand JSP does not natively support the `image=` command. Therefore a different method must be used.

As a workaround, s7ondemand offers the command vc= to send additional commands to the viewer.

The s7ondemand JSP supports a command '`vc=`' (`vc`: viewer command), which allows the viewer to accept the `image=` command. This is a valid construct:
`vc=image=S7train/shoe1,S7train/shoefront1,S7train/shoeside1`

However for this to work, the s7ondemand viewer must start by calling an image in the SKU command, any image, which will then be replaced by the string of images in the `image=` command.

Starting with a call to a standalone image:
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=Backpack_B`

And adding the `image=` viewer command to call the Image Set through the s7ondemand vc= parameter:

Dynamic Image Set (s7ondemand)
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=Backpack_B`*`&vc=image=S7train/shoe1,S7train/shoefront1,S7train/shoeside1&config=shoes`*

Issues with Zoom Targets

As said earlier, zoom targets behave differently from Image Sets to Render Sets. In Image Sets, it is assumed that each image is different (an alternate view) so zoom targets are local to each image; they do not apply to all images. With Render Sets, it is assumed that the image angle does not change, just the color or pattern, so one set of zoom targets can be applied across all images.

In the previous section, it was shown that Image and Render Sets can both be programmatically called from a URL, but zoom targets did not work as expected in the dynamic Render Set. This is because when a Render Set is created in IPS, IPS stores the zoom target information in the catalog (IPS database) line item for that Render Set. It gets that information by reading zoom target information from the individual images in the set and incorporates it into the catalog. When the viewer calls a set, it looks at this catalog field and if the zoom target information is there, it knows to process the zoom targets as a Render Set. If this information is not there, it knows to treat the zoom targets as separate items.

That information can be extracted from the Image Server using the command `req=targets`.

Here is the zoom target information for the first car image in the color_cars Render Set (S7train/Saab_red):
`http://sample.scene7.com/is/image/S7train/Saab_red?req=targets,xml`

```
<targets>
        <target frame="" label="Sporty Rear" number="1"
        rect="546,-14,610,610" userData=""/>
        <target frame="" label="Glossy Paint" number="2"
        rect="0,81,400,400" userData=""/>
        <target frame="" label="Includes Door Shadow" number="3"
        rect="217,150,400,400" userData=""/>
</targets>
```

To make this information easier to read, it has been requested as XML. Other formats supported are plain text and JSON.

The viewer command 'zoomTarget' tells the viewer to use a specific set of zoom targets in place of letting the image set its own zoom targets. The syntax for this command is:
`zoomTarget=`*index, frame, target coordinate rectangle, label*

Index is the position of the zoom target (e.g., target #1 would have an index of 1). *Frame* is the position of the image in the set; this allows zoom targets to only apply to certain images if needed. However *frame* here is blank, so if we use a value of '-1' it tells the server that these zoom targets can be applied across all images—this is key.

If the information above was parsed into this syntax, it would look like this:

```
zoomTarget=1,-1,546,-14,610,610,Sporty%20Rear;2,-
1,0,81,400,400,Glossy%20Paint;3,-1,217,150,400,400,Includes%20
Door
```

This parsing could happen on the server through an application that can parse XML or JSON, for example. The workflow would require someone on the production team to create zoom targets on one of the images, and that image could be used as the source of zoom targets, whether that image were called in the set or not.

Here is the complete dynamic Render Set using the embedded syntax, with working zoom targets across all images.

Dynamic Render Set with working zoom targets (embedded URL)

```
http://sample.scene7.com/is-viewers/flash/gener-
iczoom.swf?serverUrl=http://sample.scene7.com/is/
image/&contentRoot=http://sample.scene7.com/skins/&image=S7train/
Saab_red;S7train/red,S7train/Saab_tan;S7train/tan,S7train/
Saab_dkblue;S7train/dkblue,S7train/Saab_purple;S7train/
purple,S7train/Saab_blue;S7train/blue&config=S7train/
color_cars&viewSize=350,350&zoomTarget=1,-1,546,-
14,610,610,Sporty%20Rear;2,-1,0,81,400,400,Glossy%20Paint;3,-
1,217,150,400,400,Includes%20Door
```

Applying zoom targets to an s7ondemand viewer is more complex, because as shown earlier, the JSP cannot natively handle viewer commands, such as the `zoomTargets=` command. All viewer commands are passed through using the `vc=` command, which is specific to s7ondemand. This was done earlier to pass the `image=` command.

But to group multiple commands, the ampersands ('&') following the `vc=` command must be URL-encoded as '%26' for it to work properly. Some developers opt to URL-encode the entire string rather than just focus on the ampersands; it works either way.

Here is the group of viewer commands that must be sent to viewer to create a Render Set and see correct zoom targets:

```
image=S7train/Saab_red;S7train/red,S7train/Saab_
tan;S7train/tan,S7train/Saab_dkblue;S7train/dkblue,S7train/
Saab_purple;S7train/purple,S7train/Saab_blue;S7train/
blue&zoomTarget=1,-1,546,-14,610,610,Sporty%20Rear;2,-
```

```
1,0,81,400,400,Glossy%20Paint;3,-1,217,150,400,400,Includes%20
Door
```

Here is the same string with the ampersands ('&') URL-encoded as `%26`:

```
image=S7train/Saab_red;S7train/red,S7train/Saab_
tan;S7train/tan,S7train/Saab_dkblue;S7train/dkblue,S7train/
Saab_purple;S7train/purple,S7train/Saab_blue;S7train/
blue%26zoomTarget=1,-1,546,-14,610,610,Sporty%20Rear;2,-
1,0,81,400,400,Glossy%20Paint;3,-1,217,150,400,400,Includes%20
Door
```

And here is the entire s7ondemand URL, using the `vc=` command.

Dynamic Render Set with working zoom targets (s7ondemand)

```
http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7trai
n&sku=Backpack_B&vc=image=S7train/Saab_red;S7train/red,S7train/
Saab_tan;S7train/tan,S7train/Saab_dkblue;S7train/dkblue,S7train/
Saab_purple;S7train/purple,S7train/Saab_blue;S7train/
blue%26zoomTarget=1,-1,546,-14,610,610,Sporty%20Rear;2,-
1,0,81,400,400,Glossy%20Paint;3,-1,217,150,400,400,Includes%20
Door
```

Conclusion

Image Set and Render Sets in IPS allow clients to store image intelligence in the IPS database and not have to track it in their own content database. However, if that information needs to be responsive to sudden change or if the sheer number of images makes this approach unfeasible, then creating sets dynamically via the URL makes sense.

The method for creating Image and Render Sets from the URL is almost identical. The same viewer supports both, and can handle either a standalone image, a group of images (Image Sets), or related images and swatches (Render Sets).

Render Sets which rely on zoom targets need an extra step to make sure that the zoom targets work as expected, however if zoom targets are not needed, then that extra step is unnecessary.

Both methods are fairly straightforward if called in an embedded Flash zoom viewer, however also work on the s7ondemand (pop-up) hosted viewer environment with some modifications.

Next Steps

All of the concepts presented here are taken from the Scene7 Image Serving Viewers and Image Serving documentation, available through the Scene7 customer resource portal. For more help with integration issues and use of the Scene7 viewers, please contact Scene7 Tech Support at s7support@adobe.com.

For guided training on these concepts and consultation of best practices, please contact the Scene7 Training Department at scene7training@adobe.com.

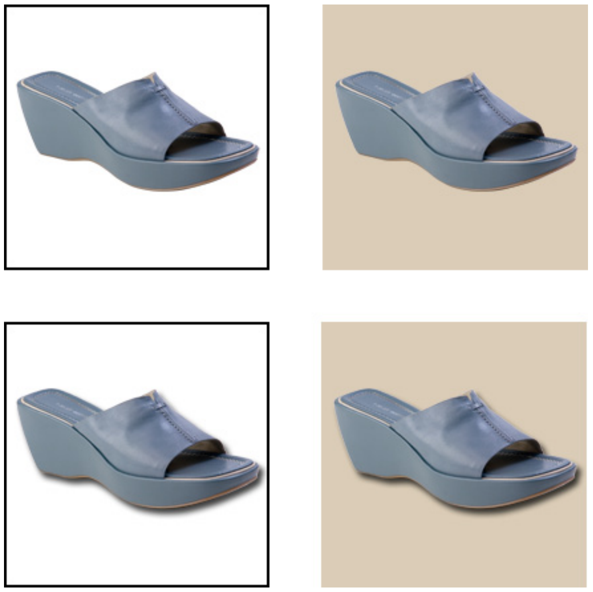
Questions? Comments?

Please contact us at ***scene7training@adobe.com*** *for additional help.*

Appendix II: Extending Image Formats with Image Modifiers

Overview

This document is designed to show graphics professionals how they can prepare their professional product photography for the most flexible output with a Scene7 image server URL construction. Content discussed here reflects a typical 'silhouette' style product photography, where a product is shown on a solid colored background, usually white.



Derivatives images created with varying URL image modifiers being passed to create background colors and/or drop shadows

Note for users of the Scene7 Publishing System (SPS)

Although this article discusses the use of IPS Image Formats, they are identical to Image Presets in SPS. All of the concepts presented here are applicable to SPS.

SPS Image Presets also have an Image Modifiers field which can be used to extend the functionality of the preset, just as it can be done with Image Formats in IPS.

Using a properly prepared source image, with a knocked out background, the image server is capable of creating a much larger array of special effects, including alternate background colors and drop shadows.

Photoshop File Preparation

In Adobe Photoshop, after you have knocked out the background so that you are seeing transparency, save the file in a fomat that respects transparency, such as a PNG, PSD or TIFF without layers but supporting transparency. (IPS/SPS does not support layered TIFF files.)

Scene7 Image Production System (IPS) Workflow

In IPS, Upload your content, making sure Mark for Publish is enabled. Publish the content to the Image Server. The content can now be accessed via a URL call to the Image Server.

Basic URL calls that generate a JPEG reply image will typically create an image on a white background. However, you may also append the URL will extra image modifiers that will act upon the image so that a special effects such as a drop shadow as well as an alternate RGB background color can be added to the image. This will work only if the source image was created with an alpha channel (transparency).

URL Syntax and Image Modifiers

Image Presets in SPS are the same thing as Image Formats in IPS. The name was changed, but the functionality and usage is the same.

For a basic understanding of Image Server URL syntax construction, please refer to the URL Construction Best Practices document. We recommend that customers access their imagery using an Image Format. An Image Format is a macro that includes image modifiers that pass information on image size, file format and sharpening parameters. These three aspects of an image are usually of chief concern to manage in an Image Format.

Often customers will employ several Image Formats, each one designed to provide a different sized image to a web site.

The syntax of a URL calling to an Image Server:

```
your path / name of image server / account name / image name /
image modifiers
```

A sample URL call passing individual image modifiers (image commands):

```
http://sample.scene7.com/is/image/S7train/9westshoe?wid=350&hei=3
50&fmt=jpeg&qlt=85,0&resMode=sharp&op_usm=0.9,1.0,8,0
```

A sample URL call replacing the image modifiers with an Image Format:

```
http://sample.scene7.com/is/image/S7train/9westshoe?$!_s7product$
```

In each case, the base URL calls to an image on an image server. The URL is broken in two by a question mark. The second part of the URL passes Image Modifiers or an Image Format (a macro pointing to a collection of image modifiers)

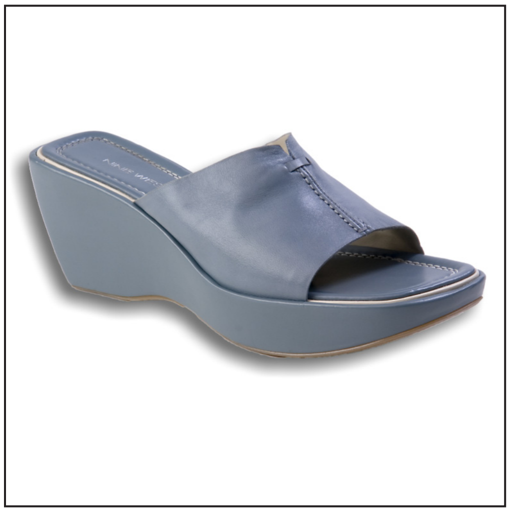
Typically Image Formats contain modifiers that control image size, image formats and sharpening aspects, as stated previously. But the Image Format creation page also allows for a user to add extra image modifiers if they wish. The following URLs expose the various image modifiers in their entirety. We suggest that, if you wish to always call images with drop shadows, you create an Image Format with the appropriate image modifier code added to the image format. This will simplify the URL.

Base image (default white background color):

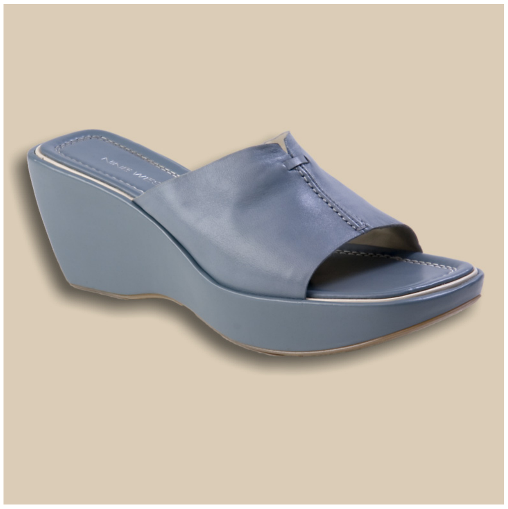
```
http://sample.scene7.com/is/image/S7train/9westshoe?wid=250&hei=2
50&fmt=jpg&qlt=85&op_usm=1,1,10,0
```

Adding a drop shadow to the image:

```
http://sample.scene7.com/is/image/S7train/9westshoe?wid=250&hei=
250&fmt=jpg&qlt=85&op_usm=1,1,10,0&layer=0&effect=-1&op_blur=40&.
effect=Drop%20Shadow&opac=55&blendmode=mult&pos=-40,40
```



Drop shadow plus a background color:
`http://sample.scene7.com/is/image/S7train/9westshoe?wid=250&hei=250&fmt=jpg&qlt=85&op_usm=1,1,10,0&layer=0&effect=-1&op_blur=40&.effect=Drop%20Shadow&opac=55&blendmode=mult&pos=-40,40&bgc=218,204,182`



Background color only, no drop shadow:
`http://sample.scene7.com/is/image/S7train/9westshoe?wid=250&hei=250&fmt=jpg&qlt=85&op_usm=1,1,10,0&bgc=218,204,182`

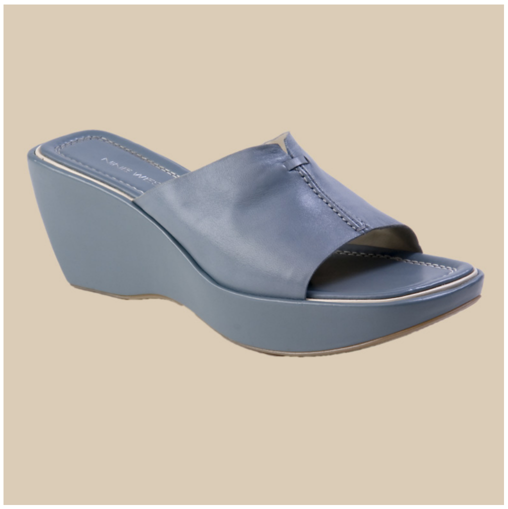
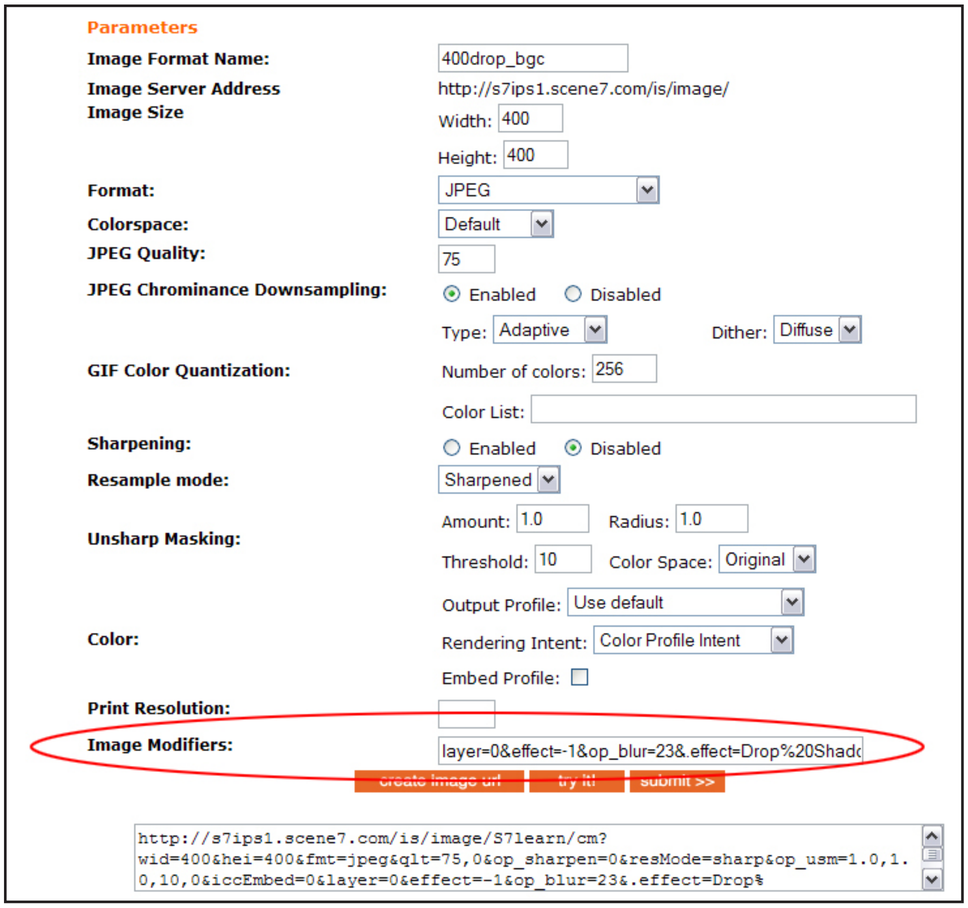


Image Formats

Typically, Image formats do not include any other special effects needed to create drop shadows and or colored backgrounds. For the complex URL below, only the image modifiers in blue are part of a typical Image Format

`http://sample.scene7.com/is/image/S7train/9westshoe?wid=250&hei=250&fmt=jpg&qlt=85&op_usm=1,1,10,0&layer=0&effect=-1&op_blur=40&.effect=Drop Shadow&opac=55&blendmode=mult&pos=-40,40&bgc=250,200,200`

You may add any other modifiers in the Image Modifiers input box at the bottom of the Image Formats page in IPS.

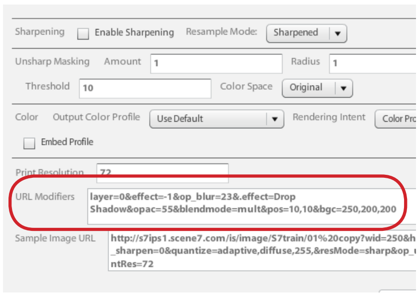


Parameters

Image Format Name:	400drop_bgc
Image Server Address	http://s7ips1.scene7.com/is/image/
Image Size	Width: 400 Height: 400
Format:	JPEG
Colorspace:	Default
JPEG Quality:	75
JPEG Chrominance Downsampling:	[✓ Enabled] Disabled
	Type: Adaptive Dither: Diffuse
GIF Color Quantization:	Number of colors: 256
	Color List:
Sharpening:	Enabled [✓ Disabled]
Resample mode:	Sharpened
Unsharp Masking:	Amount: 1.0 Radius: 1.0
	Threshold: 10 Color Space: Original
Color:	Output Profile: Use default
	Rendering Intent: Color Profile Intent
	Embed Profile: ☐
Print Resolution:	
Image Modifiers:	layer=0&effect=-1&op_blur=23&.effect=Drop%20Shad

```
http://s7ips1.scene7.com/is/image/S7learn/cm?
wid=400&hei=400&fmt=jpeg&qlt=75,0&op_sharpen=0&resMode=sharp&op_usm=1.0,1.
0,10,0&iccEmbed=0&layer=0&effect=-1&op_blur=23&.effect=Drop%
```

Image Formats page in IPS



Sharpening	☐ Enable Sharpening Resample Mode: Sharpened
Unsharp Masking	Amount 1 Radius 1
Threshold	10 Color Space Original
Color	Output Color Profile Use Default Rendering Intent Color Pro
	☐ Embed Profile
Print Resolution	72
URL Modifiers	layer=0&effect=-1&op_blur=23&.effect=Drop Shadow&opac=55&blendmode=mult&pos=10,10&bgc=250,200,200
Sample Image URL	http://s7ips1.scene7.com/is/image/S7train/01%20copy?wid=250&h _sharpen=0&quantize=adaptive,diffuse,255,&resMode=sharp&op_ ntRes=72

In SPS, Image Modifiers are found at the bottom of the Image Presets creation dialog box, called ***URL Modifiers***

By Using Image Formats, and adding any special effects in the Image Modifiers field, you may drastically shorten the URL and achieve the same effect. For instance, the following URL will create the same image with custom background color and drop shadow.

```
http://sample.scene7.com/is/image/S7train/9westshoe?$250drop_bgc$
```

An Image Format in a Scene7 Image Server URL string is accessed by the following syntax:

```
$ name of image format $
```

Appendix III: "Instant" Templates

Note for users of the Scene7 Publishing System (SPS)

Although this article discusses the use of IPS Image Formats, they are identical to Image Presets in SPS. All of the concepts presented here are applicable to SPS.

SPS Image Presets also have an Image Modifiers field which can be used to extend the functionality of the preset, just as it can be done with Image Formats in IPS.

Overview

This document is intended to show users of IPS how to create Image Formats which will mimic IPS Image Templates.

Traditional Templates vs. "Instant" Templates

Image Serving Templates are perhaps the most flexible tool in the Scene7 library. They can be used for:

- Simple image overlays, such as the ubiquitous "Ready to Publish" check mark symbol used in IPS
- Text overlays
- Multiple images with text
- Templates wrapped within other templates
- They can even be used in more exotic ways as well; the possibilities are endless

"Ready to Publish"

Text overlay

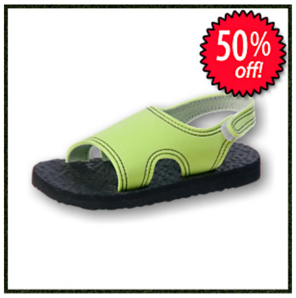
Multiple images/text

Nested templates

Image masked by text

Because templates are powerful and flexible, their URL strings tend to be more complex than ones for standard image calls. But there are ways to create layered images but keep simple URLs. Thus, the need for "instant" templates.

"Instant" templates use a sub-feature of Image Formats, called Image Modifiers. Image Modifiers allow you to type in extra URL commands, which get saved with the Image Format.



So instead of using all this code to get this image:

```
http://sample.scene7.com/is/image/S7train/
kidsandal2000?wid=300&hei=300&fmt=jpg&qlt=8
5,1&op_usm=1,1,5,0&layer=0&size=300,300&eff
ect=-1&op_blur=10&.effect=Drop%20Shadow&opac
=55&blendmode=mult&pos=5,5&layer=1&src=is{S7
train/50off?scl=1}
```

We can just use:

```
http://sample.scene7.com/is/image/S7train/
kidsandal2000?$50off$
```

Image Modifiers & You!

An Image Format is a macro that includes image modifiers that typically only controls image size, file format and sharpening parameters. But the Image Format design tool also allows for a user to add extra image modifiers if they wish.

This is a "standard" image URL in which all the modifiers are exposed:

```
http://sample.scene7.com/is/image/S7train/kidsandal2
000?layer=comp&wid=300&hei=300&fmt=jpeg&qlt=85,1&op_
sharpen=0&resMode=bicub&op_usm=1,1,5,0
```

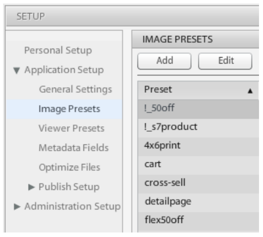
And here is a corresponding image format:

```
http://sample.scene7.com/is/image/S7train/kidsandal2000?$detail$
```

The Image Format called $detail$ instructs the Image Server to do the following:

`layer=comp`	***Tells the Image Server that the following commands affect the whole composite image, not just an individual layer***
`&wid=300&hei=300`	***Size to 300x300***
`&fmt=jpeg&qlt=85,1`	***Convert to JPEG, 85% uncompressed***
`&op_sharpen=0` `&resMode=bicub` `&op_usm=1,1,5,0`	***Bicubic resampling mode and use unsharp mask sharpening method***

In SPS, Image Presets are created under ***Setup > Application Setup > Image Presets***


SETUP

Personal Setup
▼ Application Setup
General Settings
Image Presets
Viewer Presets
Metadata Fields
Optimize Files
▶ Publish Setup
▶ Administration Setup

IMAGE PRESETS
Add | Edit

Preset ▲
!_50off
!_s7product
4x6print
cart
cross-sell
detailpage
flex50off

Each of the modifiers above, such as `&qlt=85,1`, are commands passed to the Image Server as part of the Image Format. These commands are inserted on the Image Format page:

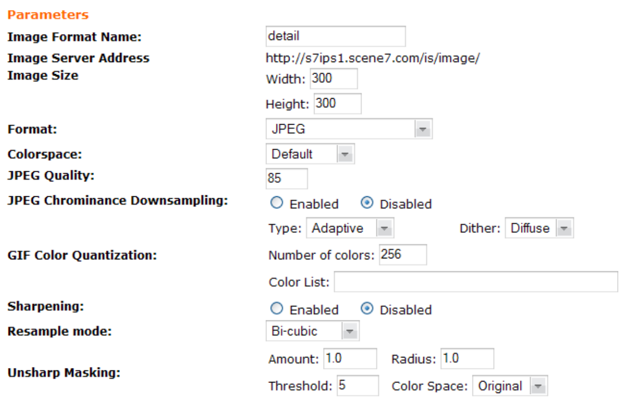
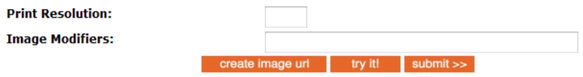
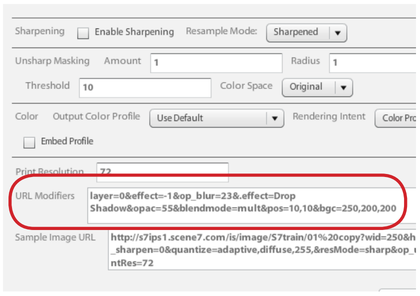

Parameters

Image Format Name:	detail
Image Server Address	http://s7ips1.scene7.com/is/image/
Image Size	Width: 300 Height: 300
Format:	JPEG
Colorspace:	Default
JPEG Quality:	85
JPEG Chrominance Downsampling:	○ Enabled ● Disabled
	Type: Adaptive Dither: Diffuse
GIF Color Quantization:	Number of colors: 256
	Color List:
Sharpening:	○ Enabled ● Disabled
Resample mode:	Bi-cubic
Unsharp Masking:	Amount: 1.0 Radius: 1.0
	Threshold: 5 Color Space: Original

At the bottom of the Image Formats page, there is a place for Image Modifiers:


Print Resolution:
Image Modifiers:


Sharpening ☐ Enable Sharpening Resample Mode: Sharpened

Unsharp Masking Amount 1 Radius 1
Threshold 10 Color Space Original

Color Output Color Profile Use Default Rendering Intent Color Pro
☐ Embed Profile

Print Resolution 72

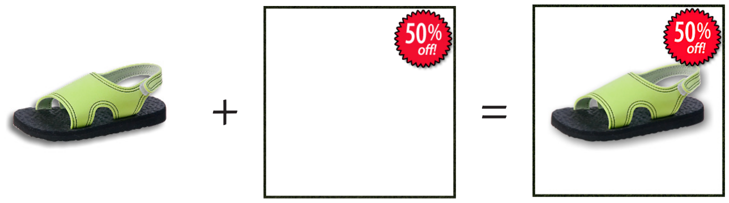
URL Modifiers layer=0&effect=-1&op_blur=23&.effect=Drop Shadow&opac=55&blendmode=mult&pos=10,10&bgc=250,200,200

Sample Image URL http://s7ips1.scene7.com/is/image/S7train/01%20copy?wid=250&h _sharpen=0&quantize=adaptive,diffuse,255,&resMode=sharp&op_ ntRes=72

In SPS, Image Modifiers are found at the bottom of the Image Presets creation dialog box, called ***URL Modifiers***

Here you can include additional commands.

URL Construction



For our template, we need to add a second layer which consists of our 300x300 pixel burst inside a frame. My burst image was uploaded to IPS and published with this URL:

`http://sample.scene7.com/is/image/S7train/50off?$!_template300$`

When using traditional Image Serving Templates, it is recommended that you create an Image Format (e.g., ***$!_template300$****) to call the template at exactly the correct size, which in this case is 300 pixels.*

In a traditional workflow, you would use the template tool to add your product image and your burst layer, save it under a single name and use parameters to switch out the product image. This would be a sample URL using the traditional template method:

```
http://sample.scene7.com/is/image/S7train/50off-template?$detail$
&$product=is{S7train/kidsandal2000?scl=1}&$burst=is{S7train/50off
?scl=1}
```

In this URL, the IPS ID being called is `50off-template` (the name of the template), and the `$burst` and `$product` parameters call the "50% off" and sandal images, respectively.

Another way to write this URL is using layering commands.

```
http://sample.scene7.com/is/image/S7train?layer=0&size=300,300&src
=is{S7train/kidsandal2000?scl=1}&layer=1&src=is{S7train/50off?scl=1}
```

Templates were created to simplify the URL, but knowing this syntax will help us build our Image Format "instant" template.

If I were to take the burst layer (layer=1; Image Server layer numbering starts at 0, the same as in Photoshop), and add it to an Image Format, I would get the following result:

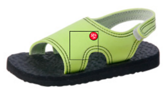


Image Modifiers: `&layer=1&src=is{S7train/50off?scl=1}`

As you can see, the burst layer is tiny compared to the main layer. Why is this?

Looking at the expanded template URL, above, you can see an extra command: `&size=300,300`. This is applied to layer 0 (the sandal layer). Why is this important? The Image Format sizes the ***composite image*** to 300x300 pixels, not the individual layers. Look again at the parameters for our original Image Format:

```
layer=comp&wid=300&hei=300&fmt=jpeg&qlt=85,1&op_
sharpen=0&resMode=bicub&op_usm=1,1,5,0
```

The command layer=comp is always part of the Image Format. It is assumed that the modifiers are to be applied to the entire, composite image. In the case of the image above, the sandal layer is much larger than the burst layer. When the burst layer is applied, it looks tiny in comparison, Then the Image Format is simply resizing that composite to 300 pixels. To override this behavior, you need to add additional commands to target layer 0 and tell it to first resize itself to 300, the exact same size as our burst layer. Therefore we add `&layer=0&size=300,300` to explicitly resize it. Here is the result:

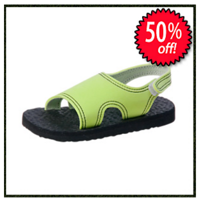


Image Modifiers: `&layer=0&size=300,300&layer=1&src=is{S7train/50off?scl=1}`

And to generate a drop shadow on layer 0, I would add this extra code:

```
&effect=-1&op_blur=10&.effect=Drop Shadow&opac=55
&blendmode=mult&pos=5,5
```

We can add a drop shadow to the sandal because it was saved on a transparent background, and uploaded to IPS as a TIFF with transparency.

For more information on this workflow and to learn how to add drop shadows and background colors to Image Formats, see Extending Image Formats with Image Modifiers (Appendix II).

Of course, your image would need to be on a transparent background for the drop shadow to appear.

Here is my final "Instant" Template Image Format and URL:

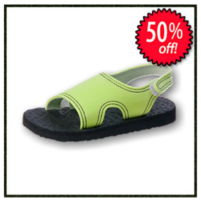
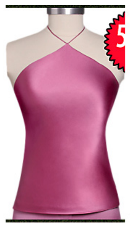


Image Modifiers: `&layer=0&size=300,300&effect=-1&op_blur=10&.effect=Drop Shadow&opac=55&blendmode=mult&pos=5,5&layer=1&src=is{S7train/50off?scl=1}`

```
http://sample.scene7.com/is/image/S7train/
kidsandal2000?$50off$
```

Mismatched Aspect Ratios

The solution above works fine when all your images have the same aspect ratio. In this case, I was applying a square burst layer to a square base image. If all my images were rectangular, then I could create a rectangular burst layer and apply it uniformly to all of my rectangular images.

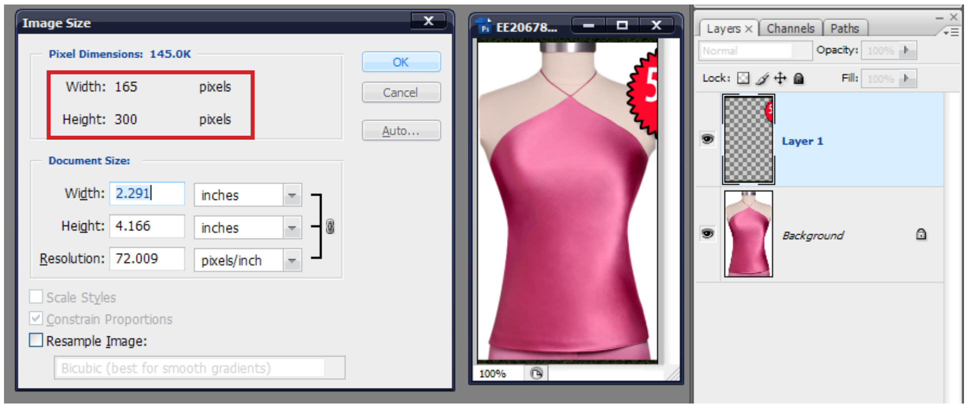


But when the base layer and overlay layer have incompatible sizes, problems occur. Watch what happens when the image format is applied to a rectangular image.

```
http://sample.scene7.com/is/image/S7train/
EE20678_16f7a?$50off$
```

This is happening because the base layer defines the overall canvas area for the entire image. So even though I am calling my composite at 300x300 and my base layer at 300x300, my imageable canvas area is only 165x300 pixels. Why is this? The Image Server will never distort the ratio of your images. So even though I'm calling it at 300x300, the maximum width it can give me at a height of 300px is 165px.

The best way to visually demonstrate what is happening is to recreate the situation using Photoshop. Layering on the Image Server mimics the behavior in Photoshop. The burst on Layer 1 is being cropped by the dimensions of Layer 0 (Background).



Pixel Dimensions: 145.0K
Width: 165 pixels
Height: 300 pixels

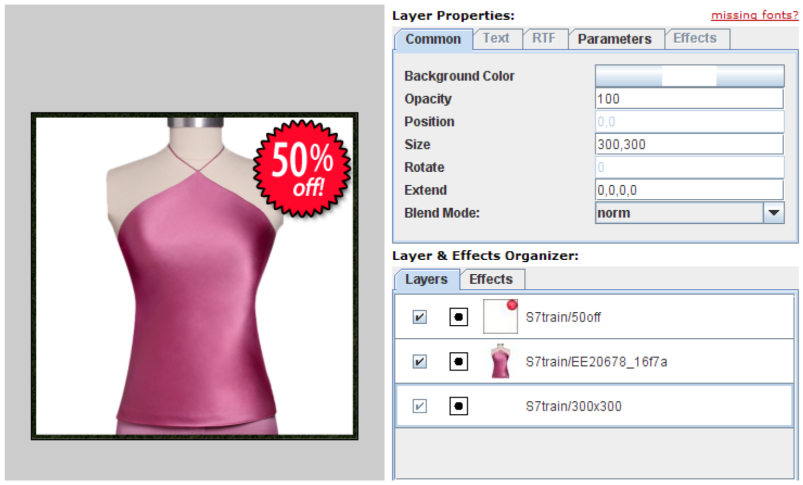
Document Size:
Width: 2.291 inches
Height: 4.166 inches
Resolution: 72.009 pixels/inch

Scale Styles
Constrain Proportions
Resample Image:
Bicubic (best for smooth gradients)

Layer 1
Background

The base layer (Background) defines the canvas in Photoshop. This is the same behavior as on the Image Server.

If we were using a traditional template, this would be no problem because we always recommend that you use a transparent base layer equal to the size of your overall template. So for every template I create, layer 0 is my blank canvas, layer 1 is the product on top of it, and layer 2 is the burst. If your base canvas layer is square, then that will work perfectly with your square overlay, even if the intermediate product layer is an odd size.



Layer Properties:

Background Color	
Opacity	100
Position	0,0
Size	300,300
Rotate	0
Extend	0,0,0,0
Blend Mode:	norm

Layer & Effects Organizer:

S7train/50off
S7train/EE20678_16f7a
S7train/300x300

The proper way to set up a traditional image template in IPS, using a blank canvas layer set to the same size as the overall template.

*To refresh your knowledge of IPS Templates, please see the **IPS Templates** series in the Training Tutorials on the Resource Center.*

But using a blank base layer is not an option when using an image format, because the base layer must be the product image. Therefore we must somehow increase the size of the base layer before the overlay layer is applied. Fortunately, such a command exists on the Image Server; it is called *Extend*. Extend adds more canvas space.

This takes a little experimentation, because Extend works with absolute pixels. So a 1000px image extended by 500 pixels would add 50% to the size, whereas it would only add 25% to a 2000px image. As an alternative, it might be more useful to use *ExtendN* instead. ExtendN (normalized Extend) uses relative values. So a value of &extendN=.5,.5,.5,.5 would add 50% to the top, bottom, left and right of the base image.

***Extend** adds margins to a layer.*

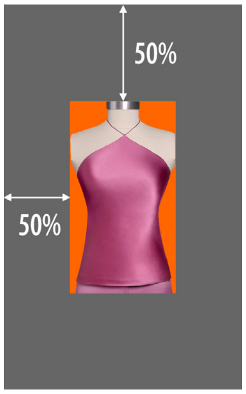
Usage:
extend = left, top, right, bottom

Number of pixels to add to the left, top, right, and bottom edge of the layer rectangle.

ExtendN *also adds to a layer.*

Usage:
extendN = leftN, topN, rightN, bottomN

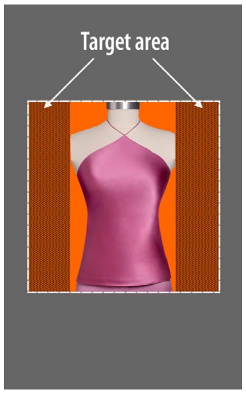
Amount of space to add to the left, top, right, and bottom edge of the layer, expressed as normalized amounts relative to the size of the original layer rectangle.



At left is my base image with an extendN command added with a value of 50%. I've filled my transparent original canvas with orange and the outer extended canvas with gray.

`http://sample.scene7.com/is/image/S7train/EE20678_16f7a?&extendN=.5,.5,.5,.5&wid=600&bgc=666666&color=ff6600`

At right, I've highlighted the desired square canvas area. You can see that a 50% extension will give us more than enough extra canvas space.



If your images have widely varying aspect ratios, use higher values of extendN.

extendN=.5,.5,.5,.5 will only work if the image's aspect ratio is in the range 2:1 through 1:2.

extendN=3,3,3,3 would increase the aspect ratio range to 4:1 - 1:4, which may be safer.

We don't need to get an exact value for the extension—we just need it to be big enough to fill out our square; the extra will be cropped off. For more extreme aspect ratios, use higher values.

Now all we have to do is call the resulting composition at a width and height of 300 pixels, and it will crop the final image. Unfortunately, it is not behaving as we expect.

Image Modifiers: `&layer=0&size=300,300&extendN=.5,.5,.5,.5&layer=1&src=is{S7train/50off?scl=1}`

The image is appearing at 300 pixels, but all the extra white space is causing our template to look too small.

scl scales the composite image by the specified scale factor.

Usage:
scl= inverse scale factor (1 or greater)

If scl= is specified, and wid= and/or hei= are present as well, the image is cropped to wid= and/or hei= after scaling.

If you've worked with templates before, you've seen scl=1 in the URL, such as `&src=is{S7train/50off?scl=1}`*. This is used to make sure that the layer being added comes in at full size before it is added, otherwise it will revert to the default image size (which is usually 400x400).*

We need to set the scale of the composite to 100%, but still crop the result at 300 pixels. Image Serving has another very useful command, `scl=1` (scale=1), which will reset the scale to a 1:1 ratio. The image will then be cropped to 300 pixels by the `wid=300&hei=300` commands.

Image Modifiers: `&layer=0&size=300,300&extendN=.5,.5,.5,.5&layer=1&src=is{S7train/50off?scl=1}&scl=1`

This gives us exactly what we need, but it would be nice to have the image appear a little smaller (so it doesn't touch our frame).

Here is our final set of modifiers, resizing the base layer to 280px, and adding the drop shadow back in. This can also be expressed as:

Image Modifiers: `&layer=0&size=280,280&extendN=.5,.5,.5,.5&effect=-1&op_blur=10&.effect=Drop%20Shadow&opac=55&blendmode=mult&pos=5,5&layer=1&src=is{S7train/50off?scl=1}&scl=1`

And here's my final URL with my new aspect ratio-neutral Image Format.

`http://sample.scene7.com/is/image/S7train/EE20678_16f7a?$flex50off$`

This concludes my Tech Tip on "Instant" Templates. To learn more about URL modifiers, check out the Image Serving documentation, found on the Client Resource Center.

Questions? Comments?

Please contact us at ***Scene7Training@adobe.com*** *for additional help.*

Appendix IV: Zooming on Templates with Flash

Overview

This section explains how you can load an Image Serving Template into a Flash zoom viewer, and still be able to change the parameters dynamically. To understand this tutorial, it is assumed that you have an understanding of templates and the Flash viewers (which means you also have a somewhat advanced understanding of IPS and Image Serving).

The Challenge of Zooming on Templates

There is nothing preventing you from calling a template in a zoom viewer, however the problem is that by definition, a template is a dynamically changeable image and cannot be defined by a single name. When calling to a static image, you simply load the name of the image into the zoom viewer. However doing the same thing with a template will load only the template default values.

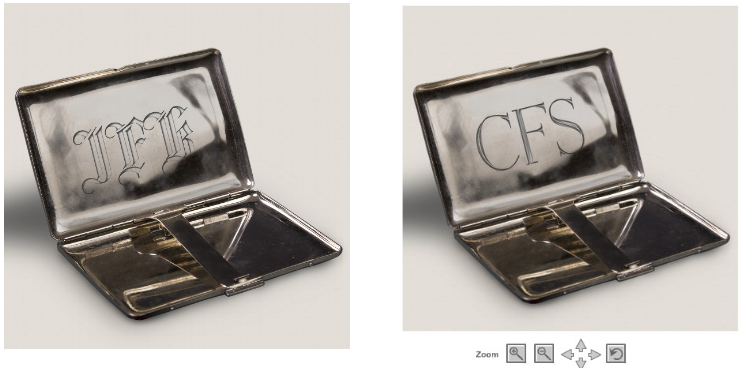
Here is a static image URL:
`http://sample.scene7.com/is/image/S7train/adobe_sneaker?$!_s7product$`

And here is the same image loaded in a zoom viewer:
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=adobe_sneaker`

Static image called as a JPEG, and in a zoom viewer. Images are the same

This is a template called as a static JPEG:
`http://sample.scene7.com/is/image/S7train/case-engrave?$!_s7product$&$font=Fancy Card Text&$txt=JFK`

And here we'll load it into a zoom viewer:
`http://sample.scene7.com/s7/zoom/flasht_zoom.jsp?company=S7train&sku=case-engrave`



Template called as a JPEG, and in a zoom viewer. Images are different because parameters are missing

The results are different for the template. Why is this? The answer lies in the URL.

***Sample* is not your URL!**

The URL *sample.scene7.com* is for training purposes only.

Use your assigned production URL given in the Scene7 Welcome Letter.

It will typically be in the format, e.g., `http://s7d5.scene7.com`.

This base URL should be at the beginning of **every** Scene7 URL call.

And origin URLs (in the form of testvipd5.scene7.com) are for QA and testing only.

When in doubt, check your Welcome Letter or contact Tech Support (s7support@adobe.com)

In the static version, there are a couple of parameters that specify the text (`$txt=JFK`) and the font face (`$font=Fancy Card Text`). In the zoom version, clearly the text is "CFS," and the font is different, but no parameters are present.

If you pass no parameters to a template or if nothing has changed, the template will revert to its default values

This is because the zoom version is calling the ***default values***. This template was originally set up using that particular font and with "CFS" as the default text. If you need to see it differently, you change the relevant parameter. But if you leave off the parameters, the template reverts to its default state, i.e., the version that was originally authored.

Passing Parameters to the Zoom Viewer

So *why* the template is not appearing as needed is clear—the parameters are missing. This should be obvious to anyone who is familiar with templates. But *how* to add the parameters is less obvious.

Can we simply put them at the end of the viewer URL? (Hint: No.)

```
http://sample.scene7.com/s7/zoom/flasht_zoom.
jsp?company=S7train&sku=case-engrave&$font=Fancy%20Card%20
Text&$txt=JFK
```

Template parameters only work on the Image Server. Image Serving Viewers, which have a different command set, do not support parameters directly

You can try this, but it won't help. Template parameters are relevant to the Image Server, not the viewers, and act as commands which tell the Image Server to do something. The parameters are meaningless to the viewers, so they simply ignore them.

Parameters cannot be passed directly on the URL. However Image Server commands can be passed *indirectly* through the viewers by appending them as modifiers to the the `image=` viewer command. Sending commands to the viewer is also discussed in Appendix I: "Dynamic Image & Render Sets".

Image Command Syntax

If you are familiar with the embedded Flash viewer code, you are already using the `image` command, however you might only be calling to an image without modifiers. The basic syntax for the `image` command is an image name followed by a string of modifiers separated by ampersands (&):

```
& image = Company Name / IPSID ? modifier1 & modifier2 ...
```

However since the viewers cannot distinguish between Image Server commands and regular viewer commands (because both use ampersands and other similar characters), the modifier commands must be URL encoded (specifically, all occurrences of '&' and '=' in must be encoded as %26 and %3D, respectively). Therefore an image command with modifiers would typically look like this:

```
&image=S7train/adobe_sneaker%3Fresmode%3Dsharp2%26op_usm%3D1,1,8
```

This tells the viewer to serve up the adobe_sneaker image and to use the "sharp2" resample method as well as add some unsharp masking. Note that the '?' (%3F), '=' (%3D) and '&' (%26) operators are all encoded. As a best practice, **always URL-encode your viewer image string**.

Calling Template URL as an Image String

Now we have the methodology for calling a template URL as an image string. Using the cigarette case example, the image string would look like this in plain text:

```
&image=S7train/case-engrave?$font=Fancy Card Text&$txt=JFK
```

And look like this encoded:

```
&image=S7train/case-engrave%3F%24font%3DFancy%20Card%20
Text%26%24txt%3DJFK
```

Notice that I removed the sharpening commands. While it is true that you need to add sharpening—the viewers will <u>not</u> sharpen your images unless you tell them to—you would

As a best practice, create a single zoom configuration/preset and call that for every zoom viewer.

typically append these as modifiers to the viewer itself rather than on the image directly. More common, though, you would add the sharpening to your viewer configuration in IPS, and add those settings using the `config=config name` command. As a best practice, always call a viewer configuration, especially if you are calling to an embedded viewer.

Calling a Template in an Embedded Zoom Viewer

Using the embedded viewer, we would then pass the whole image string as an argument (sharpening is added using the modifier= command and a viewSize is added too).

Zooming on a template (embedded URL):

```
http://sample.scene7.com/is-viewers/flash/genericzoom.
swf?serverUrl=http://sample.scene7.com/is/image/&image=S7train/
case-engrave%3F%24font%3DFancy%20Card%20Text%26%24txt%3DJFK&modif
ier=resmode=sharp%26op_usm=1,1,8&viewSize=500,475
```

Technically, the `image=` and `modifier=` commands can both pass modifiers. You can also append your sharpening commands to the image command, just as you can put all the template parameters using the modifier command. However the image command is required to call at least the base image; modifiers are optional. We'll use the modifier command when passing parameters to the s7ondemand viewers.

Calling a Template in an s7ondemand Zoom Viewer

In Dynamic Image & Render Sets (Appendix I), we used viewer commands (`vc=`) to pass additional commands to the viewer. We will use that technique here to send the template parameters to the zoom viewer.

What is s7ondemand?
s7ondemand is a viewer platform consisting of pre-configured Java Server Pages (JSP), hosted on the Scene7 On-Demand network.

It simplifies calling zoom, spin, and eCatalog products by pre-configuring and embedding the Flash viewer in a web page, and exposing the least number of parameters required by the viewer.

The main difference in our method is that s7ondemand viewers do not support the `image=` command directly, but instead use the `sku=` command to pass a single image name to the viewer with no modifiers. So this syntax is **illegal** on s7ondemand:

```
& sku = Company Name / IPSID ? modifier1 & modifier2 ...
```

Therefore, as in our earlier s7ondemand example, this will be our base URL:

```
http://sample.scene7.com/s7/zoom/flasht_zoom.
jsp?company=S7train&sku=case-engrave ...&more to come
```

Next, we construct our viewer command. First, we assemble all of our parameters as image modifiers:

```
modifier=$font=Fancy Card Text&$txt=JFK
```

If we want to add sharpening here, we can add them to the end of the string:

```
modifier=$font=Fancy Card Text&$txt=JFK&resmode=sharp2&op_
usm=1,1,8
```

Why all the encoding?
We already know that the embedded viewer requires encoding because it allows the viewer to tell apart Image Serving commands from viewer commands.

s7ondemand requires double encoding because in reality we are wrapping the modifiers inside a proxy command (vc). Technically, this then separates other viewer commands from the Image Server modifier string. (phew!)

Bookmark a good URL Encoder/Decoder on your web browser for testing. I use this one:
`http://meyerweb.com/eric/tools/dencoder/`

Now, to make this understandable by s7ondemand, we have to **double** URL-encode the modifiers (encoded twice). For example, every ampersand (&) will be encoded as %2526.

In reality, you can get away with double-encoding only the ampersands, but in practice it may be easier to encode the entire string at once. This is the modifier string double encoded:

```
modifier=%2524font%253DFancy%2520Card%2520Text%2526%2524txt%253DJ
FK%2526resmode%253Dsharp2%2526op_usm%253D1%252C1%252C8
```

We are now ready to add the modifier to the URL string, however s7ondemand does not let you add viewer commands directly; instead you wrap them in a proxy command called `vc` (viewer command).

```
&vc=modifier=%2524font%253DFancy%2520Card%2520Text%2526%2524txt%2
53DJFK%2526resmode%253Dsharp2%2526op_usm%253D1%252C1%252C8
```

Lastly, we put the entire URL together.

Zooming on a template (s7ondemand URL):

```
http://sample.scene7.com/s7/zoom/flasht_zoom.
jsp?company=S7train&sku=case-engrave&vc=modifier=%2524font%253DFa
ncy%2520Card%2520Text%2526%2524txt%253DJFK%2526resmode%253Dsharp2
%2526op_usm%253D1%252C1%252C8
```

Conclusion

In this article, I discussed the challenges of zooming on a template, and the two methods of implementing zoom—using an embedded Flash viewer and using an s7ondemand pop-up Flash viewer.

The embedded method is slightly more straight-forward than using s7ondemand as less URL encoding is necessary, however based on your implementation and hosting environment, using the embedded viewers may not be an option.

Also realize that either method involves a fair amount of front-end development. We recommend that if the material found in this article is difficult to comprehend or if your require additional resources, you contact Adobe and arrange for a custom development and/or consultation quote from the Adobe Scene7 Professional Services team. Alternatively, you can arrange an advanced URL construction seminar with the Training Department.

For help with integration issues and use of the Scene7 viewers, or for further documentation, please contact Adobe Scene7 Technical Support at s7support@adobe.com.

For guided training on these concepts and consultation of best practices, please contact the Adobe Scene7 Training Department at scene7training@adobe.com.

Questions? Comments?

Please contact us at
scene7training@adobe.com
for additional help.

Appendix V: Zooming on Vignettes with Flash

Overview

This article explains how you can load an Adobe Scene7 rendered image (vignette) into a Flash zoom viewer. To understand this tutorial, it is assumed that you understand Image Rendering URL construction and have a basic knowledge of the Flash viewers.

Vignettes (also referred to by their file format extension, VNT) are dynamic document files that respond to commands from URLs. Unlike static JPEGs and TIFFs, VNTs can be rendered in realtime to create derivative versions of themselves, using the compiled masks, illumination maps and flowlines that are created with the Image Authoring application. To handle these special files, a dedicated Render Server has to handle the rendering and communication with the VNT; because this is such a specialized and CPU-intensive function, this server is separate from the Image Server.

The Adobe Scene7 viewers can natively handle images from the Image Server only—rendered images have to be loaded in a special way

The Challenge of Zooming on Vignettes

Unlike Image Serving Templates (see *Appendix IV: "Zooming on Templates with Flash"*), vignettes and rendered images are not recognized by the viewers at all. Templates, although special in their own right, are products of the Image Server. The viewers can natively handle Image Serving images *only*, not Image Rendering images (vignettes). Therefore the entire VNT and all of its commands must be loaded into the zoom viewer by another method—compare that with templates, which can be directly loaded in the viewer, but have its parameters loaded indirectly.

*You can set the default image size for the Render Server in IPS by going to **Publish > Publishing Settings (for Scene7 Image Rendering) > Default View Size**.*

Keep in mind this can be no larger than the Max Image Height and Width (also found on the same page).

*In SPS, the same settings are found under **Setup > Application Setup > Publish Setup > Image Renderer**.*

Pyramid Vignettes and You!

Up until recently, (i.e., in earlier versions of IPS and Image Rendering) zooming on vignettes was not possible at all. VNTs were published as 'single resolution' files, whereby if you wanted to have your rendered image in multiple sizes, you had to explicity publish it in multiple sizes. (See the sidebar, "What happened to Vignette Formats?" in the URL Construction Best Practices, "Vignettes & Render Server URLs" on page 7).

But the Render Server evolved, and can now publish out vignettes with multiple resolutions "baked in." So just as IPS can create Pyramid TIFFs, it can also now create Pyramid Vignettes. Having multiple sizes allows efficient resizing and, to the point of this article, zooming.

Preparing the URL

Before we can talk about zooming, we need a usable render URL. Since the size will be set by the zoom, we do not need to provide a size on the URL. For now, the Render Server will give the image back to us according to our default publish settings. (For most accounts, this is 100px. Yeah, that's really lame. Please complain to Tech Support—I know I have.) Here is a sample URL:

```
http://sample.scene7.com/ir/render/
S7trainRender/chairwithlayers?obj=chair
&src=camo_chettah_45ppi&res=25&illum=1&
obj=welting&color=138,8,17
```

This calls the ever-popular chair, applies a cheetah print and colorizes the welting. I'm not adding sharpening here, because I plan to add it to the zoom viewer, although I could if I needed extra sharpening.

Sample is not your URL!

The URL *sample.scene7.com* is for training purposes only.

Use your assigned production URL given in the Scene7 Welcome Letter.

It will typically be in the format, e.g., `http://s7d5.scene7.com`.

This base URL should be at the beginning of **every** Scene7 URL call.

And origin URLs (in the form of testvipd5.scene7.com) are for QA and testing only.

When in doubt, check your Welcome Letter or contact Tech Support (s7support@adobe.com)

Going to the Src (Source)

Here is where we need to get creative in our workflow.
As I said earlier, the zoom viewer cannot accept a rendered image *directly*, but what about indirectly? In *Zooming on Templates with Flash (Appendix IV)* and *Dynamic Image & Render*

Sets (Appendix I) articles, we used the `image=` and `modifier=` command to pass additional information to the viewers. This technique is similar, except we are passing an entirely different image via the Image Server command `src=` (source).

The source command is native to both the Image and Render Server, but here we are using it in its Image Serving context. The basic usage is:

```
src = IPSID
```

This is fine as long as the image is coming from the same server and account, but to call it from a different server or different account we need to use the extended syntax:

```
src = [is | ir] { CompanyName / IPSID ? modifiers }
```

This tells the server whether the image is coming from the Image Server (`is`) or the Render Server (`ir`). This allows us to combine images from the Image and Render Server, and is the heart of the workflow for dynamic personalization (e.g., calling an Image Serving text template and applying it to a vignette—see sidebar, "Zooming on Personalization."). Since we need to pass an image from the Render Server to the zoom viewer via the Image Server, this is the way we'd do it with our rendered chair example:

```
src=ir{S7trainRender/chairwithlayers?obj=chair&src=camo_chettah_4
5ppi&res=25&illum=1&obj=welting&color=138,8,17}
```

We now have an image URL string that the Image Server will understand. But how do we apply it to the viewer?

Zooming on Personalization

The workflow for zooming on personalized vignettes (i.e. vignettes that call to an Image Serving Template as a decal, rather than a tiling texture), is exactly the same as it is for zooming on regular vignettes (described in this article).

The same amount rules of URL-encoding certain commands and parameters apply, however it gets even more complicated with personalized renders because of the number of URL parameters.

Before you approach this, make sure you understand zooming on templates (see *Appendix IV*) and zooming on vignettes (this article).

We recommend you contact Training for more specialized help in this area (scene7training@adobe.com).

Bait and Switch

As said before, we cannot load the rendered image directly because the zoom viewer will not recognize it. Yet we need to call to *some* image or else nothing will happen. Depending on your implementation method (embedded zoom or s7ondemand pop-up zoom) the method is slightly different, however in essence we swap out a static image with the rendered image as the viewer loads. In other words, we call the viewer and tell it to load an image from the Image Server, but then instead call to the image in our `src` command.

Let's look at how this technique is used for both the embedded Flash zoom viewer and the s7ondemand zoom viewer.

Loading the Vignette in the Viewer (Embedded Method)

We'll start with a sample zoom viewer URL that loads a placeholder image (Backpack_A):

```
http://sample.scene7.com/is-viewers/flash/genericzoom.
swf?serverUrl=http://sample.scene7.com/is/image/&image=S7train/
Backpack_A&modifier=sharp2%26op_usm=1,1,8&viewSize=500,475
```

To learn more about viewer modifiers see Dynamic Image & Render Sets (Appendix I) and Zooming on Templates with Flash (Appendix IV)

I've highlighted the placeholder image because we will be replacing that in a moment. I've also added an image modifier for sharpening as well as a command to size the overall viewer.

Notice that the modifier command has a URL-encoded '&' (%26) to separate the `resmode=` *and* `op_usm` *commands*

We can replace the placeholder image in this way:

```
&image=S7train?src=ir{S7trainRender/chairwithlayers?obj=chair&src
=camo_chettah_45ppi&res=25&illum=1&obj=welting&color=138,8,17}
```

Even though we are not explictly calling to an image (just the S7train company), this is a valid Image Serving request. Here is a static, non-zooming example:

As a best practice, sharpening is not added as a modifier, but called as part of a zoom viewer configuration, and called with `config=[Config name]`

```
http://sample.scene7.com/is/image/S7train?src=ir{S7trainRender/
chairwithlayers?obj=chair&src=camo_chettah_45ppi&res=25&illum=1&o
bj=welting&color=138,8,17}
```

This produces essentially the same image as the base Render Server URL example, except now it is being re-routed through the Image Server. But we are not ready to add it to the viewer yet, because first we need to encode the string. In *"Zooming on Templates with Flash"*, I explain that these requests need to be URL-encoded or else the viewer will get confused as to what are viewer commands (like size and sharpening) and what are image commands. A very good URL Encoder can be found at `http://meyerweb.com/eric/tools/dencoder/`

Here is the same image URL string after being encoded:

```
&image=S7train%3Fsrc%3Dir%7BS7trainRender/chairwithlayers%3Fqlt%3
D80%26obj%3Dchair%26src%3Dcamo_chettah_45ppi%26res%3D25%26illum%3
D1%26obj%3Dwelting%26color%3D138%2C8%2C17%7D
```

I'll replace the placeholder image with my encoded image string to get my final URL.

Zooming on a vignette (embedded URL):

```
http://sample.scene7.com/is-viewers/flash/genericzoom.
swf?serverUrl=http://sample.scene7.com/is/image/&image=S7tra
in%3Fsrc%3Dir%7BS7trainRender/chairwithlayers%3Fobj%3Dchair%
26src%3Dcamo_chettah_45ppi%26res%3D25%26illum%3D1%26obj%3Dwe
lting%26color%3D138%2C8%2C17%7D&modifier=resmode=sharp2%26op_
usm=1,1,8&viewSize=500,475
```

What is s7ondemand?
s7ondemand is a viewer platform consisting of pre-configured Java Server Pages (JSP), hosted on the Scene7 On-Demand network.

It simplifies calling zoom, spin, and eCatalog products by pre-configuring and embedding the Flash viewer in a web page, and exposing the least number of parameters required by the viewer.

Loading the Vignette in the Viewer (s7ondemand Method)

Calling to the rendered image in the s7ondemand viewer is a similar process, but a little more complex. This is because the s7ondemand viewer is optimized for simple requests and has fewer manual API controls than the embedded viewers.

As with the embedded viewer, we first form a zoom URL to a placeholder image:

```
http://sample.scene7.com/s7ondemand/zoom/flasht_zoom.jsp?company=
S7train&sku=Backpack_A
```

This time, the placeholder image has to remain part of the URL because we cannot use the same method as above. The `sku=` command expects a single image ID; it will not accept an image string in the same way that the `image=` command will.

Therefore we must pass in the render string using the `vc=` command (viewer command), combined with either `image=` or `modifier=`. The s7ondemand commands `company=` and `sku=` are internally translated to the embedded command `image=` in this way:

```
image = company / sku
```

By supplying an additional `image=` command, we are overwriting the initial `company=` and `sku=` commands. The s7ondemand string would look like this:

```
& vc = image = ir { Image Rendering String }
```

Why all the encoding?
We already know that the embedded viewer requires encoding because it allows the viewer to tell apart Image Serving commands from viewer commands.

s7ondemand requires double encoding because in reality we are wrapping the modifiers inside a proxy command (vc). Technically, this then separates other viewer commands from the Image Server modifier string. (phew!)

Bookmark a good URL Encoder/Decoder on your web browser for testing. I use this one:
`http://meyerweb.com/eric/tools/dencoder/`

We will append the same `image=` string to our s7ondemand URL that we used for the embedded URL:

```
image=S7train?src=ir{S7trainRender/chairwithlayers?obj=chair&src=
camo_chettah_45ppi&res=25&illum=1&obj=welting&color=138,8,17}
```

As before, we will need to encode it before adding to the URL, so that the Image Server can distinguish it apart from the other viewer commands. If you read *"Zooming on Templates with Flash"*, you learned that image strings passed using the `vc=` command must be **double** URL-encoded, which is why this method is more complicated than using the embedded URL. For example, every ampersand (&) will be encoded as %2526 instead of %26.

However, the commands `image=` and `modifier=` are native viewer commands, therefore those commands themselves must be **single** URL-encoded to separate them from the Image Server commands they contain. After URL encoding the `image=` command **once** and URL encoding the image string **twice**, the image URL will look like this:

```
no encoding _ | encoded once| _________________________encoded twice_______________________________ |
      &vc=image%3DS7train%253Fsrc%253Dir%257BS7trainRender/cha
```

```
irwithlayers%253Fobj%253Dchair%2526src%253Dcamo_chettah_
45ppi%2526res%253D25%2526illum%253D1%2526obj%253Dwelting
%2526color%253D138%252C8%252C17%257D
```

If we need to add additional parameters (like sharpening), we'd typically add them in the viewer configuration (config=), or can append as double-encoded viewer modifiers, but separated by single URL-encoded ampersands (& = %26).

Here is the modifier string (sharpening commands):

```
modifier=resmode=sharp2&op_usm=1,1,8
```

Following the `modifier=` command, we must double-encode the commands, just as we did with the `image=` string. But the ampersand (&) separating the modifiers from the `image=` command must be single-encoded. Here is the outline of the encoded `vc=` syntax:

```
_____no encoding____|__ encoded once ___|___________ encoded twice ___________ | _ encoded once __ |
     & vc =    command %3D [ command %2526 command ] %26
               command %3D [ command %2526 command ]
```

Note that the `vc=` *command itself is preceded by an unencoded ampersand (&). This is important because it is a native s7ondemand command.*

Some of the most common native s7ondemand commands are `company=`, `sku=`, `config=`, `vc=`, `zoomwidth=` *and* `zoomheight=`.

Some embedded viewer commands include image=, modifier=, viewSize= and skin=. These commands are all documented in

For additional s7ondemand commands, contact Tech Support at s7support@adobe.com.

Here is the `vc=` string with additional commands:

```
&vc=image%3DS7train%253Fsrc%253Dir%257BS7trainRender/chairwit
hlayers%253Fobj%253Dchair%2526src%253Dcamo_chettah_45ppi%2526
res%253D25%2526illum%253D1%2526obj%253Dwelting%2526color%253D
138%252C8%252C17%257D%26modifier%3Dresmode%253Dsharp2%2526op_
usm%253D1%252C1%252C8
```

Zooming on a vignette (s7ondemand URL):

```
http://sample.scene7.com/s7ondemand/zoom/flash_zoom.jsp?company=
S7train&sku=Backpack_A&vc=image%3DS7train%253Fsrc%253Dir%257BS7tr
ainRender/chairwithlayers%253Fobj%253Dchair%2526src%253Dcamo_chet
tah_45ppi%2526res%253D25%2526illum%253D1%2526obj%253Dwelting%2526
color%253D138%252C8%252C17%257D%26modifier%3Dresmode%253Dsharp2%2
526op_usm%253D1%252C1%252C8
```

Therefore, unless you really need to use s7ondemand for this application, I would highly recommend you use the embedded zoom viewer due to the complex encoding requirements. Again, the primary use of s7ondemand is for simple URL calls, but clearly zooming on a vignette is not a simple URL call.

Conclusion

In this article, I discussed the challenges of zooming on a vignette, and the two methods of implementing zoom—using an embedded Flash viewer and using an s7ondemand pop-up Flash viewer.

The embedded method is more straight-forward than using s7ondemand as much less URL encoding is necessary, however based on your implementation and hosting environment, using the embedded viewers may not be an option.

Also realize that either method involves a fair amount of front-end development. We recommend that if the material found in this article is difficult to comprehend or if your require additional resources, you contact Adobe and arrange for a custom development and/or consultation quote from the Adobe Scene7 Professional Services team. Alternatively, you can arrange an advanced URL construction seminar with the Training Department.

For help with integration issues and use of the Scene7 viewers, or for further documentation, please contact Adobe Scene7 Technical Support at s7support@adobe.com.

For guided training on these concepts and consultation of best practices, please contact the Adobe Scene7 Training Department at scene7training@adobe.com.

Questions? Comments?

Please contact us at ***scene7training@adobe.com*** *for additional help.*