How-To Implement Image Optimization

Overview

StackPath's image optimization service allows your images to be optimized by compressing, resizing, cropping and converting them without compromising the quality of the image. This feature enhances image quality across multiple devices, while decreasing image size, resulting in faster load times, which in turn can improve a websites' overall SEO.

Note: This solution is in Limited Availability, meaning that current features and capabilities are subject to change.

This document will provide you with instructions for how to implement StackPath's image optimization service on your site using the JavaScript **Resolve** method**.**

Please contact your Sales Representative to enable Image Optimization on your account. They will provide you with a custom domain used to serve your images.

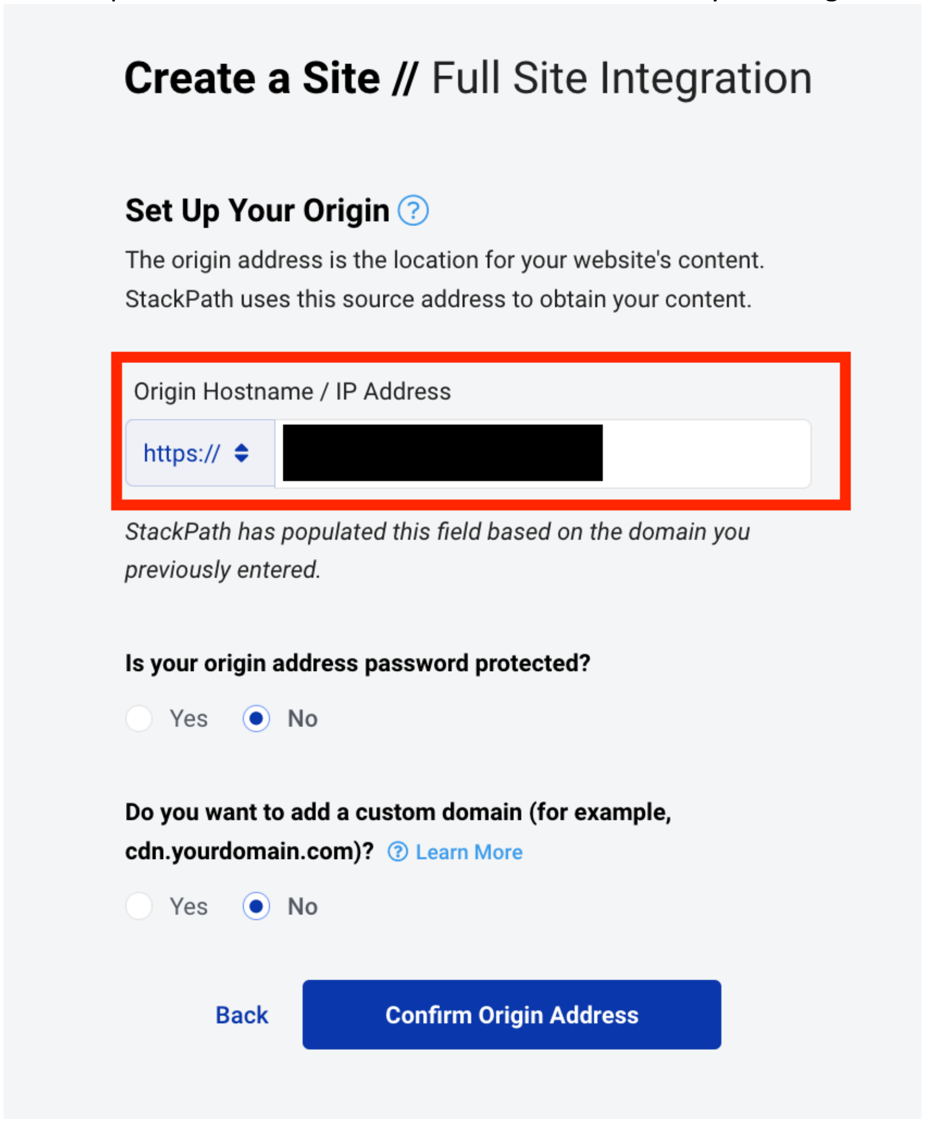
CDN Configuration

This section demonstrates how to create a separate, fully-integrated StackPath CDN site, parallel to your original site, that's dedicated to serving your optimized images, providing full control over the caching policies.

Please follow the steps below for instructions on how to create and configure this second CDN site:

1. Navigate to the **Stack** in which you would like to place this Site.
2. From the Stack Dashboard, in the left-side navigation, click **Sites**.
3. Click **Create Site**.
4. Click **Select Full Site**.
5. Enter a domain name for the subdomain from which your optimized images will be served, select **CDN** services, then click **Set Up Your Origin**. In our example below, we created a subdomain named images.domain.com, that will serve optimized images for our main site, domain.com

6. In the **Origin Hostname / IP Address** field, enter the domain you received from your Sales Representative. This is the domain that will host your images.



Create a Site // Full Site Integration

Set Up Your Origin

The origin address is the location for your website's content. StackPath uses this source address to obtain your content.

Origin Hostname / IP Address

https:// [redacted]

StackPath has populated this field based on the domain you previously entered.

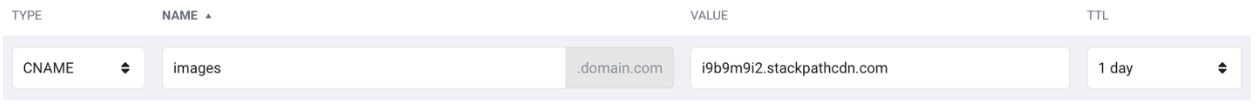
Is your origin address password protected?

Yes [✓ No]

Do you want to add a custom domain (for example, cdn.yourdomain.com)? Learn More

Yes [✓ No]

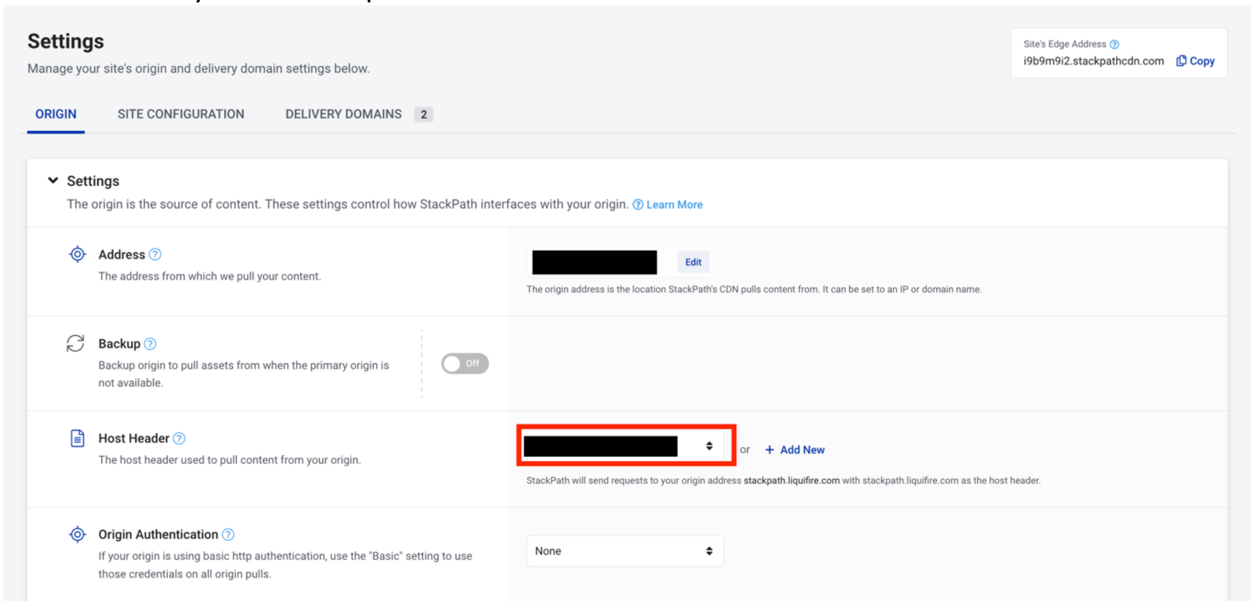
7. Follow the on-screen instructions for how to set up an SSL certificate. When you're finished, click **Confirm SSL Method**.
8. Navigate to your website's managed DNS page. Create a CNAME record for the subdomain that you created that will serve your optimized images and point it to the Edge Address shown on the screen. When you're finished, click **Complete Setup**.



TYPE	NAME ▲	VALUE	TTL
CNAME	images .domain.com	i9b9m9i2.stackpathcdn.com	1 day

9. After successfully creating your second CDN site, navigate to the site's **Settings** tab in the left-side navigation menu and change the **Host Header** to the same domain you

received from your Sales Representative.



Settings

Manage your site's origin and delivery domain settings below.

Site's Edge Address
i9b9m9i2.stackpathcdn.com Copy

ORIGIN | SITE CONFIGURATION | DELIVERY DOMAINS 2

Settings

The origin is the source of content. These settings control how StackPath interfaces with your origin. Learn More

Address
The address from which we pull your content.

[redacted] Edit

The origin address is the location StackPath's CDN pulls content from. It can be set to an IP or domain name.

Backup
Backup origin to pull assets from when the primary origin is not available.

Off

Host Header
The host header used to pull content from your origin.

[redacted] or + Add New

StackPath will send requests to your origin address **stackpath.liquifire.com** with stackpath.liquifire.com as the host header.

Origin Authentication
If your origin is using basic http authentication, use the "Basic" setting to use those credentials on all origin pulls.

None

This completes the StackPath and CDN portion of the setup. Please see the sections below for the next steps, which involves making changes to your website's HTML/source code.

Implementing Resolve

StackPath's image optimization service uses a JavaScript method named **Resolve**, which is the method behind the automatic optimization and adaptation of images.

The steps below describe how to implement **Resolve** on your site.

Step 1: Import the JavaScript Library and Create an Instance of Resolve

In order to implement **Resolve**, you'll have to import a JavaScript library and create an instance of **Resolve**. This JavaScript library is important, as it provides the constructor needed to implement **Resolve**.

The script below, which goes in the <head> section of your site's HTML, automatically imports the JavaScript library for you and creates an instance of **Resolve**. It contains the following:

- Constructor: com.liquidpixels.Resolve(options)
- Server: images.domain.com (this is the subdomain that you created)
- URI: stackpath

```
<head>
<script src='http://images.domain.com/zap/dhtml/com.liquidpixels.Resolve.jsr' type='text/JavaScript'> </script>
<script type='text/JavaScript'>
var resolve = new com.liquidpixels.Resolve({
server: "images.domain.com",
uri: "stackpath"
});
</script>
</head>
```

Step 2: Add a <div> Element Containing an <img> Tag that Sources a Blank Image or Image Chain

This step prevents certain browsers from displaying a broken image icon upon initial page load.

Within the `<body>` element of your site's HTML, add a `<div>` element where you will specify the server name, URI, height and width (in pixels) of your original image. Note that we are still using `images.domain.com` as the source in the `<img>` tag:

```
<body>
<div>
<img src='http://images.domain.com/stackpath?blank=height[1],width[1],color[none]&
sink=format[gif]' height='height' width='width'/>
</div>
</body>
```

Step 3: Add the Resolve Attributes to the <img> Tag and Specify an Image or Image Chain

Within the same `<img>` tag that we created in step 2, add the **Resolve** attributes that identify the class and image or image chain.

For this example, the source of our image is `http://domain.com/images/image.jpg`.

To specify an image, use the *data-resolvesrc* attribute:

```
<body>
<div>
<img src='http://images.domain.com/stackpath?blank=height[1],width[1],color[none]&
sink=format[gif]' height='height' width='width' class='Resolve' data-resolvesrc='http://domain.com/images/image.jpg' />
</div>
</body>
```

To specify an image chain, use the *data-resolvechain* attribute:

```
<body>
<div>
<img src='http://images.domain.com/stackpath?blank=height[1],width[1],color[none]& sink=format[gif]' height='height'
width='width' class='Resolve' data-resolvechain='source=url[file: image.jpg]&sink' />
</div>
</body>
```

Step 4: Putting it all Together

If you put together steps 1 through 3, your HTML file should resemble the following:

```
<!doctype html>
<html lang='en'>
<head>
<script src='http://images.domain.com/zap/dhtml/com.liquidpixels.Resolve.jsr' type='text/JavaScript'>
</script>
<script type='text/JavaScript'>
var resolve = new com.liquidpixels.Resolve({
server: "images.domain.com",
uri: "stackpath"
});
</script>
<title>Title</title>
</head>
<body>
<div>
<img src='http://images.domain.com/stackpath?blank=height[1],width[1],color[none]&
sink=format[gif]' height='height' width='width' class='Resolve' data-resolvesrc='http://domain.com/images/image.jpg' />
</div>
</body>
</html>
```

Step 5: Verify the Image Displays Properly

Open the HTML file in a web browser to verify that your image(s) was processed properly. Use your browser's developer tools to inspect the element. If you see that the class attribute is *ResolveComplete*, then the image was successfully processed using **Resolve**.

```
<div>
<img src='http://images.domain.com/stackpath?blank=height[1],width[1],color[none]&
sink=format[gif]' height='height' width='width' class='ResolveComplete' data-
resolveSrc='http://domain.com/images/image.jpg' id='uic_0' />
</div>
```

JavaScript API Callback Functions

JavaScript API callback functions return information about the **Resolve** instance or the image and browser. These functions are to be added to the **options** object when creating an instance of **Resolve**.

When adding a callback function, the syntax `<callback_name>: function(arg1, arg2)` should be used, followed by code that handles the returned data.

In the example below, the code uses the `handleEvent` callback function to display an alert if **Resolve** returns an error. The `handleEvent` callback function provides information about the status of the **Resolve** instance.

```
handleEvent: function(response, data) { if (response == 'error') {
alert(response + ': ' + data.message;
}
```

}

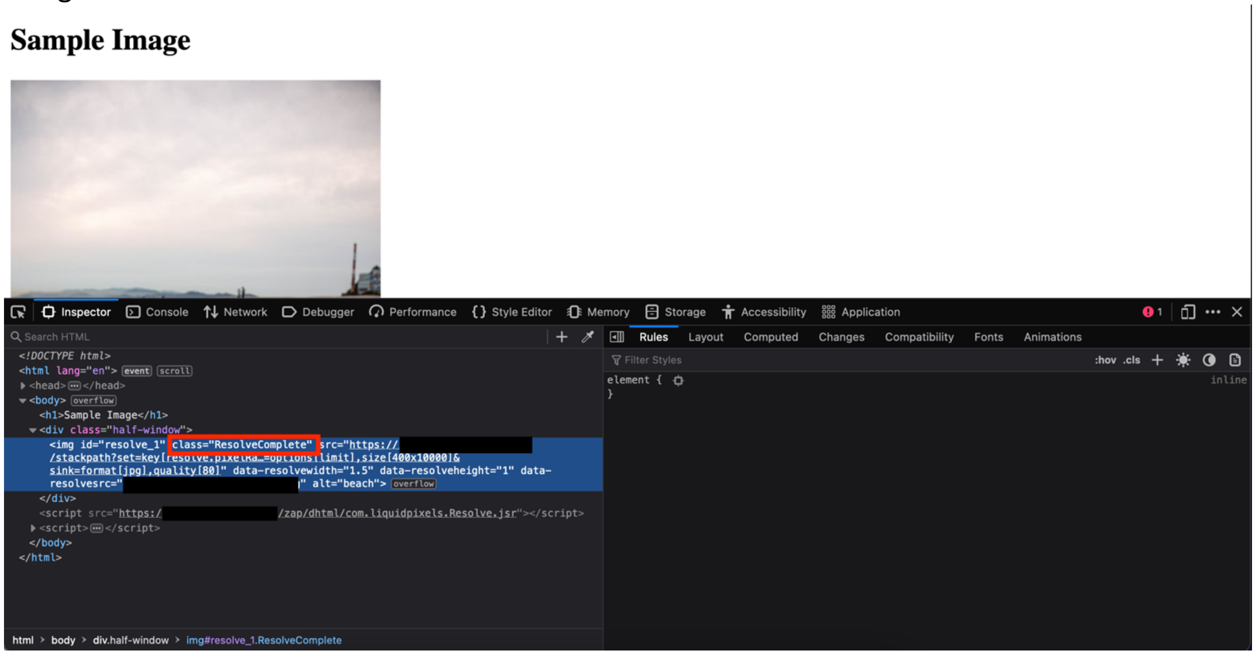
Image Optimization Sample Script

In the example HTML below, we are using the *output* function within our instance of **Resolve** to customize an image's output quality based on the type of device used to load it.

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>StackPath Image Optimization Example</title>
</head>

<body>

<h1>Sample Image</h1>
<div class="half-window">
<img class="Resolve" src="https://images.domain.com/stackpath?blank=width[1],height[1]&sink=format[gif]" data-
resolvewidth="1.5" data-resolveheight="1" data-resolvesrc="https://domain.com/beach.jpg"
alt="beach"></img>
</div>

<script src="https://images.domain.com/zap/dhtml/com.liquidpixels.Resolve.jsr"></script>
<script>
var resolve = new com.liquidpixels.Resolve({
server: "images.domain.com",
uri: "stackpath",
imageFit: "containerWidth",
output: function(browser, image) {
console.log(browser, image);
var quality = 80;
var pixelRatio = window.devicePixelRatio;

if (browser.platform.type && browser.platform.type.match(/mobile/)) {
quality = 70;
pixelRatio = 1;
}
return {
quality: quality,
pixelRatio: pixelRatio
};
},
handleEvent: function(type, data) {
console.log("[%o]: %o", type, data);
}
});

</script>
</body>

</html>
```

We can verify if this script is correct and processing the image properly by using our browser's developer tools.

In the screenshot below, we opened this HTML file in our browser, inspected the element using the "Inspector" tool , and found that the class attribute is equal to *ResolveComplete*, which tells us that the image was successfully processed
using **Resolve.**



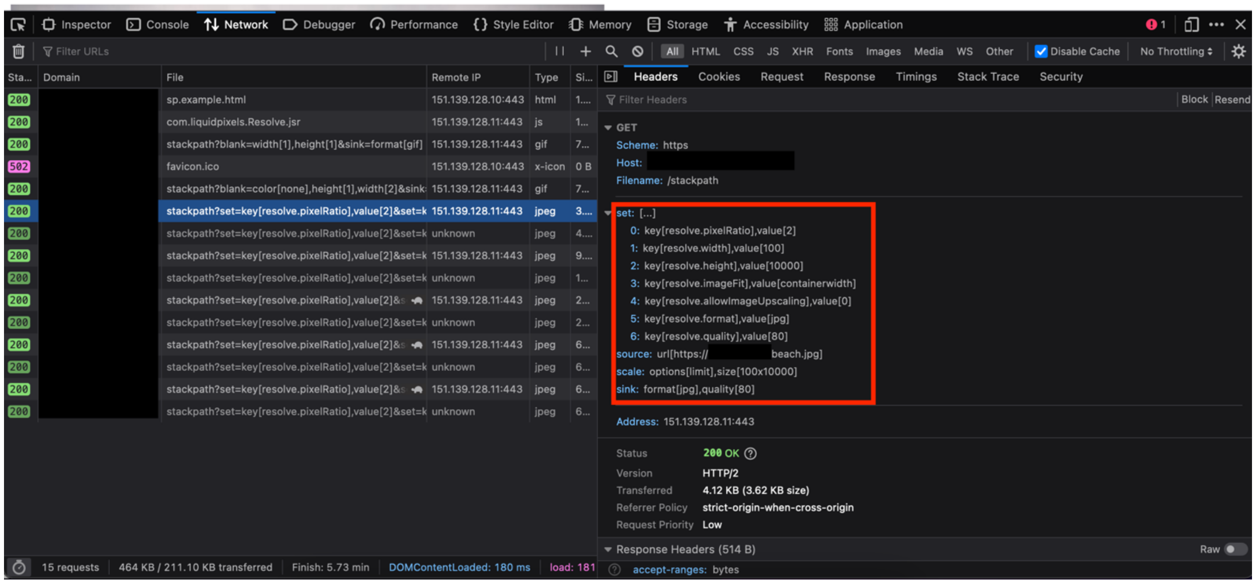
```
<!DOCTYPE html>
<html lang="en"> event scroll
▸ <head>⋯</head>
▾ <body> overflow
    <h1>Sample Image</h1>
  ▾ <div class="half-window">
      <img id="resolve_1" class="ResolveComplete" src="https://[redacted]
      /stackpath?set=key[resolve.pixelRa…=options[limit],size[400x10000]&
      sink=format[jpg],quality[80]" data-resolvewidth="1.5" data-resolveheight="1" data-
      resolvesrc="[redacted]" alt="beach"> overflow
    </div>
    <script src="https:/[redacted]/zap/dhtml/com.liquidpixels.Resolve.jsr"></script>
  ▸ <script>⋯</script>
  </body>
</html>
```

```
element {
}
```
inline

html › body › div.half-window › img#resolve_1.ResolveComplete

Within the same browser inspection window, if we select the "Network" tab, then select the image, we can see all of the functions that we defined in our script were applied to the image, meaning it was optimized as expected.

Sample Image



Inspector | Console | Network | Debugger | Performance | Style Editor | Memory | Storage | Accessibility | Application

Filter URLs | All | HTML | CSS | JS | XHR | Fonts | Images | Media | WS | Other | ✓ Disable Cache | No Throttling

Sta...	Domain	File	Remote IP	Type	Si...
200	[redacted]	sp.example.html	151.139.128.10:443	html	1....
200		com.liquidpixels.Resolve.jsr	151.139.128.11:443	js	1...
200		stackpath?blank=width[1],height[1]&sink=format[gif]	151.139.128.11:443	gif	7...
502		favicon.ico	151.139.128.10:443	x-icon	0 B
200		stackpath?blank=color[none],height[1],width[2]&sink	151.139.128.11:443	gif	7...
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	151.139.128.11:443	jpeg	3....
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	unknown	jpeg	4....
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	151.139.128.11:443	jpeg	9....
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	unknown	jpeg	1...
200		stackpath?set=key[resolve.pixelRatio],value[2]&s	151.139.128.11:443	jpeg	2...
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	unknown	jpeg	2...
200		stackpath?set=key[resolve.pixelRatio],value[2]&s	151.139.128.11:443	jpeg	6...
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	unknown	jpeg	6...
200		stackpath?set=key[resolve.pixelRatio],value[2]&s	151.139.128.11:443	jpeg	6...
200		stackpath?set=key[resolve.pixelRatio],value[2]&set=k	unknown	jpeg	6...

15 requests | 464 KB / 211.10 KB transferred | Finish: 5.73 min | DOMContentLoaded: 180 ms | load: 181

Headers | Cookies | Request | Response | Timings | Stack Trace | Security

```
GET
  Scheme: https
  Host: [redacted]
  Filename: /stackpath

set: [...]
  0: key[resolve.pixelRatio],value[2]
  1: key[resolve.width],value[100]
  2: key[resolve.height],value[10000]
  3: key[resolve.imageFit],value[containerwidth]
  4: key[resolve.allowImageUpscaling],value[0]
  5: key[resolve.format],value[jpg]
  6: key[resolve.quality],value[80]
source: url[https://[redacted]beach.jpg]
scale: options[limit],size[100x10000]
sink: format[jpg],quality[80]

Address: 151.139.128.11:443

Status            200 OK
Version           HTTP/2
Transferred       4.12 KB (3.62 KB size)
Referrer Policy   strict-origin-when-cross-origin
Request Priority  Low

Response Headers (514 B)
accept-ranges: bytes
```

The *output* function is one of many functions you can use to customize **Resolve**. To view more function types and customizations, please see our Image Optimization Reference Guide.

Troubleshooting

Issue	Probable Cause	Possible Solution
The image or image chain is not displaying in the browser. The page is blank.	The **enabledClass** field specified a new value when the **Resolve** instance was created, but the <img> tag refers to the default class of 'Resolve'.	Make sure the class attribute in the <img> element is set to the same value set in the **enabledClass** field of the **Resolve advancedOptions** object. The image or image chain does not display if these values are not the same.
	The web page cannot communicate with StackPath's image optimization service.	Make sure the server and URI fields are correct when creating the **Resolve** object.
	The image chain is missing the **sink** command. Without the **sink** command, **Resolve** cannot process the image chain.	Add the **sink** command to the end of the image chain referenced in *data-resolvechain*.
	The image was created using the **newImage** (attributes) JavaScript API method, but the width and height were not specified. Without the width	Either add the width and height values to the attributes object or update the image using the **updateImage (element) updateImages (imgArray)** method.

	and height, **Resolve** is unable to process the image.	
The image or image chain is not displaying in the browser. The page shows a broken image icon.	The image or image chain does not exist in the location specified by the *data-resolvesrc* or *data-resolvechain* attribute.	Verify the image or image chain resides in the correct folder. Also, make sure the base location is valid if specifying the base field when creating the **Resolve** object.
GIF images are not loading.	**Resolve** is attempting to scale a GIF image to a high resolution, or the GIF image has several frames.	Set the **ignoreGIF** field in the **advancedOptions** JavaScript object to **true**. This is especially important if scaling up GIF images with multiple frames, which can be resource intensive