Handling Local Images

Textbox.io gives you the ability to handle local images in one of several ways within your application. You may either upload local images from the client to your application, store images directly in the editor generated HTML itself (using base64 data URIs), or have the Textbox.io editor prevent local images from being inserted.

With all but the last option (prevent local images) the user experience is the same: users can add images to Textbox.io instances via the image upload dialog, by dragging and dropping images from their computer, or via copy-paste.

Image Handling Option	Description
Keep image data in HTML content	[Default] Local images are stored within the editor's HTML content as base64 encoded data URIs.
Upload images	Local images are uploaded to a remote server when added to the editor via HTTP POST. Textbox.io automatically updates the `<image>` `src` attribute with the new path to the uploaded image. See the information on uploading local images below to learn how to configure Textbox.io to do this.
Prevent local image insertion	Local image functionality is turned off - users can no longer use the local image upload dialog tab, drag-drop, or copy paste to add images to editor content. See the information on Preventing Local Image Insertion below to learn how to configure Textbox.io to do this.

Local vs. Remote Images

Local images are defined as those residing on the client filesystem. They may also be part of a word processor document or otherwise present on the clipboard. Textbox.io can be configured to upload local images to your application or embed them in editor HTML.

Remote images are those which exist on a remote host and are accessible via a URL.

Storing Local Images in Content (base64 data URIs)

Textbox.io will by default store local images added to an editor as embedded base64 data URIs.

If this is the desired editor behavior in your application, no further action is necessary. When a user adds an image to a Textbox.io editor within your application textbox.io will automatically embed that image into the HTML content.

Uploading Local Images

Configuring your application and Textbox.io for local image uploads involves first creating a server-side handler and then configuring your Textbox.io instance to use that handler.

Image Upload on Form Submission

If you have enabled the image upload functionality of Textbox.io is it ***strongly*** recommended that you review the information on Handling Asynchronous Image Uploads.

Textbox.io uploads images asynchronously to ensure the author's flow isn't interrupted by multiple image upload dialogs/prompts. However, developers need to be mindful of this when integrating Textbox.io so as to ensure all images are uploaded prior to content being submitted to a server.

Server-side Upload Handler

In order to upload local files to the remote server via HTTP POST, you will need a server-side upload handler script that accepts the images and objects on the server and stores them in the correct directory or database. This script is the same script that would be used for uploading any file to the server via the HTTP POST method.

For example, when you use a file input element (`<INPUT type="file">`), the script specified in the `action` attribute of the parent `<form>` element is used to upload the file to the server.

The server-side upload handler script should return a JSON object similar to the one below. The returned JSON should include a single location attribute with the path to the stored image as it's value.

```
{ "location" : "/uploaded/image/path/image.png" }
```

Note, that the '/' here in the `location` field is used to suggest a `root-relative` path. If you don't provide the leading '/', then the path will be `relative`. If you provide a protocol (e.g. `http`), then the path will be `absolute`. The table below shows how the full image path will be resolved against the various types of image locations:

Base Path	Image Location	Path Type	Full Image Path
http://server-name/base/	/uploaded/image.png	root-relative	http://server-name/uploaded/image.png
http://server-name/base/	uploaded/image.png	relative	http://server-name/base/uploaded/image.png
http://server-name/base/	http://elsewhere/image.png	absolute	http://elsewhere/image.png

Example Upload Handler Scripts

The following scripts are reference implementations for handling server-side uploads with Textbox.io. Please note that these scripts are provided only for reference - they are not intended for production use.

- Node.js Upload Handler
- PHP Upload Handler

Your upload handler script should:

- Store the image in a location appropriate for your application
- Success: Return JSON with the path to the uploaded image
- Failure: Return HTTP 500 if an error occurs

Configuring Textbox.io to Use a Server-side Upload Handler

Once you've set up a server side upload handler, all that's left is to make Textbox.io aware of the handler's location via a `configuration` object and set up the path used to construct the `<image> src` attribute.

In the example below, any local image added to a the Textbox.io editor is uploaded to the handler script located at ttp://example.com/postAcceptor.php. Textbox.io then constructs the path to the newly uploaded image by combining the (optional) `basePath` value and the image filename. The resulting editor html is then updated with the new path to the image, generating HTML like: `<img src="/my/application/images/filename.png" />`.

```
var config = {
        images : {
                upload : {
                        url : '/postAcceptor.php',          // Handler URL
                        basePath: '/my/application/images/',                        // Remote image storage path
                        credentials: false                                                      //
Optional: sends cookies with the request when true
                }
        }
};

var editor = textboxio.replace('#targetId', config);
```

For more detail on see the images configuration property.

CORS Considerations

You may choose for your web application to upload image data to a separate domain. If so, you will need to configure Cross-origin resource sharing (CORS) for your application to comply with JavaScript "same origin" restrictions.

CORS has very strict rules about what constitutes a cross-origin request. The browser can require CORS headers when uploading to the same server the editor is hosted on, for example:

- A different port on the same domain name
- Using the host IP address instead of the domain name
- Swapping between HTTP and HTTPS for the page and the upload script

The upload script URL origin must *exactly* match the origin of the URL in the address bar, or the browser will require CORS headers to access it. A good way to guarantee this is to use a relative URL to specify the script address, instead of an absolute one.

All supported browsers will print a message to the JavaScript console if there is a CORS error, and Textbox.io will display an error banner.

The Reference Upload Handler Scripts provided here configure CORS on a per script basis. You may also choose to configure CORS at the web application layer or the HTTP server layer.

Further Reading on CORS

- W3C Wiki - CORS Enabled
- MDN - HTTP access control (CORS)
- W3C - Cross-Origin Resource Sharing Specification

Preventing Local Image Insertion

If you wish to prevent the insertion of local images by users you may do so by setting the `images.allowLocal` to `false`.

When `images.allowLocal` is set to false, users will be unable to add local images to editor content from the local image upload dialog tab, by drag-drop, or by copy paste. If a user takes an action that would normally result in the insertion of an image, a notification will be displayed that insertion of images is not allowed.

The example below creates an editor where local images have been prevented.

```
var config = {
        images : {
                allowLocal : false        // Prevent users from adding local images
        }
};

var editor = textboxio.replace('#targetId', config);
```