Premium Web Documentation (Client-Side High Resolution)

With our High Resolution photo API you can edit photos of any size directly in your browser. We accomplish this by exposing a **saveHiRes**() method on our standard **Aviary.Feather** object. When you call **saveHiRes**(), a list of the actions applied to the photo in the editor are automagically sent to our servers along with a url, that you supplied, to the original resolution photo for rendering. You can override the default **save**() action by passing an **onSaveButtonClicked** function to the launch configuration like so:

```
var editor = Aviary.Feather({
        //...
        onSaveButtonClicked: function() {
            //  Important Signature's must be unique for each
editor.saveHiRes() call.
            //  See Authentication section below.
            editor.saveHiRes();
            return false;
        }
});
```

You'll need to pass some additional authentication parameters parameters to the editor's launch configuration object to make this work.

Additional Required Parameters

- **apiKey**: [string] get your API key by creating a new web app here.
- **salt**: [string] randomly generated string. This must be unique for every save. We recommend a GUID. See Signature Generation Example below.
- **timestamp**: [string] a UNIX timestamp used to validate the signature. This will be the number of seconds since the EPOCH.
- **signature**: [string] see Authentication
- **encryptionMethod**: [string] supports **sha1**, **md5** or **sha256**
- **hiresUrl**: [string] a url pointing to the original resolution image.
- **onSaveHiRes**: [function(domId, url)] a function that's called when rendering is complete.

Here are some additional **optional** parameters:

Optional Parameters

- **backgroundColor**: [string] use to specify the background color when an image with transparent pixels is converted to a JPG. Default: "0xFFFFFF"
- **fileType**: [string] specify the file type of the output image. Default: tries to parse extension from hiresUrl.
- **jpgQuality**:[integer] use when saving a JPEG. Default: 100
- **hiresWidth**: [integer] pixel width of the high resolution photo. This will let a user track the size of her photo in the lower left corner of the editor.
- **hiresHeight**: [integer] pixel height of the high resolution photo. This will let

a user track the size of her photo in the lower left corner of the editor.

- **hiresMaxSize**: [integer] max pixel width and height of the high resolution image output. Cap's the resize tool, and when used in conjunction with *hiresWidth* and *hiresHeight*, will allow the user to track the ultimate size of her photo in the lower left corner of the editor.

Example

```
var editor = Aviary.Feather({
    apiKey: 'mykey',
    timestamp: '1358129513',
    salt: 'MfXgvL',
    encryptionMethod: 'sha1',
    signature: 'f64e20af04076bbf4a1171cffb1c69ce89493fff'
    hiresUrl: ‘https://www.google.com/images/srpr/logo3w.png',
});
```

You can also pass these parameters when you launch the editor.

Example

```
var editor.launch(
    apiKey: 'mykey',
    timestamp: '1358129513',
    salt: 'MfXgvL',
    encryptionMethod: 'sha1',
    signature: 'f64e20af04076bbf4a1171cffb1c69ce89493fff'
    hiresUrl: ‘https://www.google.com/images/srpr/logo3w.png',
});
```

Authentication

All high resolution renders require the use of a **signature**. Signatures are generated by concatinating your **api_key**, **api_secret**, the current UNIX **timestamp**, and a randomly generated **salt**. The resulting string is then hashed into a signature using one of the following encryption methods: md5, sha1, or sha256. The encryption method used must be sent with all requests.

Example

```
signature = md5(api_key+api_secret+timestamp+salt)
```

If your signature isn't valid, the onError callback will be called with details.

Example

```
var editor = Aviary.Feather({
    ...
    onError: function(code, msg) {
        alert(code);
    }
});
```

Important: Signature's must be unique for *each* **saveHiRes()** call. This means that if you intend to allow user's to save multiple times in a session, you'll need to generate a new signature on your server (via an AJAX call) and update the editor configuration with the required parameters before re-enabling the editor. You can accomplish this with the **updateConfig(obj)** method call.

Example

```
// call this BEFORE calling saveHires() for a second time
function updateSignature(callback)  {
    $.ajax('http://localhost:3000/genSignature', function(data) {
        editor.updateConfig({
            apiKey: data.apikey,
            salt: data.salt,
            timestamp: data.timestamp,
            signature: data.signature
        });

        // safe to call saveHires() now
        callback();
    });
}
```

Note: Make sure your /generateSignature endpoint authenticates your user so that this endpoint can't be hijacked allowing hires saves using your apiKey.

Signature Generation Examples

Below you'll find an example of generating a valid signature on the server.

```
var uuid    = require('node-uuid');
var crypto  = require('crypto');

var getAuth = function(apikey, apisecret, method) {
    var getSalt = function() {
        return uuid.v4(); // random id
```

```
    };

    var getUnixTimestamp = function() {
        return (new Date()).getTime() / 1000;
    };

    var getSignature = function(key, secret, timestamp, salt,
encryptionmethod) {
        // Concat string
        var sig = key + secret + timestamp + salt;

        // Encrypt
        return
crypto.createHash(encryptionmethod).update(sig).digest('hex');
    };

    var authObj = {
        apiKey: apikey,
        salt: getSalt(),
        timestamp: getUnixTimestamp().toString(),
        encryptionMethod: method
    };

    authObj.signature = getSignature(authObj.apiKey, apisecret,
authObj.timestamp,  authObj.salt, authObj.encryptionMethod);

    return authObj;
}
```