Blackinton Design-A-Badge IFRAME/HTTP-POST Integration

Overview

This document contains instructions for integrating the Blackinton Design-A-Badge web technology into a dealer website. Integration is facilitated using the *Blackinton Design-A-Badge Integration Extensions* with IFRAME user interface integration and the HTTP-POST data return method.

User Interface Integration Method
IFRAME

Data Return Method
HTTP-POST

Design-A-Badge Process Completed Action
HTTP Redirection

It is important to note that the Design-A-Badge Integration Extensions relies on a small server-side footprint. The server-side code, which will be contained within the "dealer-site", is offered in Classic ASP, ASP.NET (C#), and PHP (release 5.2.0).

There is also a small client-side footprint. This code must be included within the web page that will host the Design-A-Badge web application or the IFRAME. This code is composed of pure JavaScript. It is within this code that the Design-A-Badge web application can be customized to meet the need of the dealer, simply by adjusting declared variable values.

Each of these components is explained in detail within the following sections.

Download all code needed for this integration method using the link below.

http://www.blackinton.com/integrate/support/support.zip

Client Side JavaScript

Supporting client side JavaScript code can be found within a file named “head.js”. This code must be copied from the file and placed directly into the page that will host the <IFRAME>. The JavaScript code contains remote scripting tags, so it cannot be placed in a remote script tag itself. You must cut and paste this code in to the hosting page.

Minor modifications must be made to the code after it is copied into the hosting page in preparation for use. These changes are summarized below.

The lines of code that must be changed are found at the top of the “head.js” file. It is also important that only the values indicated below are modified. When making the changes read the instructions closely, incorrect modification may result in runtime script errors and could lead to unexpected results.

dealerId:

```
var dealerId   = '44453XX';
```

This variable must contain your valid dealer identification number. This is a value provided to you previously by Blackinton and is the same number used to access the dealer portal.

_switches:

```
var _switches = '10000000'
```

This variable works in a binary fashion (feature dip-switch), where each position within the string must contain a ‘1’ or a ‘0’. Each position within the variable corresponds to a specification application feature that can be turned on or off.

A value of ‘1’ indicates that the feature represented by that character position is ENABLED or ON. Conversely, a value of ‘0’ indicates that the feature is DISABLED or OFF.

The feature/character position table below summarizes the presently available features.

POSITION	FEATURE
1	**Pricing.** **This feature will render the pricing section at the bottom of the design canvas on the "design-a-badge.php" page. Pricing is based on Blackinton suggested retail pricing. The pricing is controlled by Blackinton.** **0=on screen price is not enabled, 1=on screen pricing is enabled**
2	**Rendering Footprint.** **This feature will determine if surrounding content will be shown or suppressed. Content would include the header and left region that surround the core interactive portion of Design-A-Badge interface.** **0=surrounding content is enable, 1=disabled**
3	**HTTP POST Data Format.** **This feature will determine how the data posted by the DAB service module will be returned to the dealer website. This does not affect the HTTP post, just the contents of the POST.** **This will also shorten the steps of the entire Design-A-Badge process to 3. There is no contact information step or confirmation step. This is best used when a shopping cart integration feel is desired.** **1=Conventional POST. All fields, including the Transaction Number are individual HTML encoded fields.**
4	**QTY Field will be rendered on the Design Canvas when this switch is set to "1".** **This feature will render at QTY field on the Design Canvas, enabling the web visitor to specify how a QTY. If this switch is set to "0", then a QTY of 1 is transmitted.** **1=QTY Field is rendered.**
5	**Make "Address" field on the Contact Information page a required field. When set to "1", this field cannot be blank. By default the "Address" field is not required.**

6	**Reply-To Address on emails.** **When set to "1", the REPLY-TO address field of the email sent to the customer will be the value of the _emailTo field (see above), and the REPLY-TO address field of the email sent to the dealer will be the value of the email address specified by the customer on the Contact Information step.** **The FROM address on emails sent from Blackinton will be:** ***Design-A-Badge@blackinton.com*** **When set to "0", the FROM address will be:** **noreply@blackinton.com**
7	**Reserved for future features, should be set to 0**
8	**Reserved for future features, should be set to 0**

_postOnLoad:

var _postOnLoad = false;

This is a very important setting, which depending on the value, will determine when the HTTP-POST will occur. The first choice is to have the HTTP-POST occur as soon as the final page of the Design-A-Badge process displays (this is the ThankYou.php page). Alternatively, the post can occur when the web visitor clicks the EXIT button (or if you change the button face text to "Add to Cart", for example).

If the goal is to implement a shopping cart solution, then _postOnLoad should be set to 'false'. If the goal is just to log the design information into a database, for example, then the value should be set to 'true'; this will guarantee the HTTP-POST back occurs even if the "EXIT" button is never clicked by the web visitor. If _postOnLoad is set to 'false', failure to click the "EXIT" button will mean no HTTP-POST.

_disableEmail:

var _disableEmail = true;

This variable enables or disables the sending of an email to recipients of the dealer. When using the HTTP POST data return method, email is normally disabled, though it is possible to use the email method at the same time. By default, this value is set to 'true', meaning the email will be disabled.

If this option is set to 'false', enabling email sending, then the dealer email address on file with Blackinton will receive an email message with design details. Additionally, email addresses specified in the ***_emailTo*** variable will also receive the same email (see below).

_emailTo:

var _emailTo = 'user@domain.com;user2@domain.com';

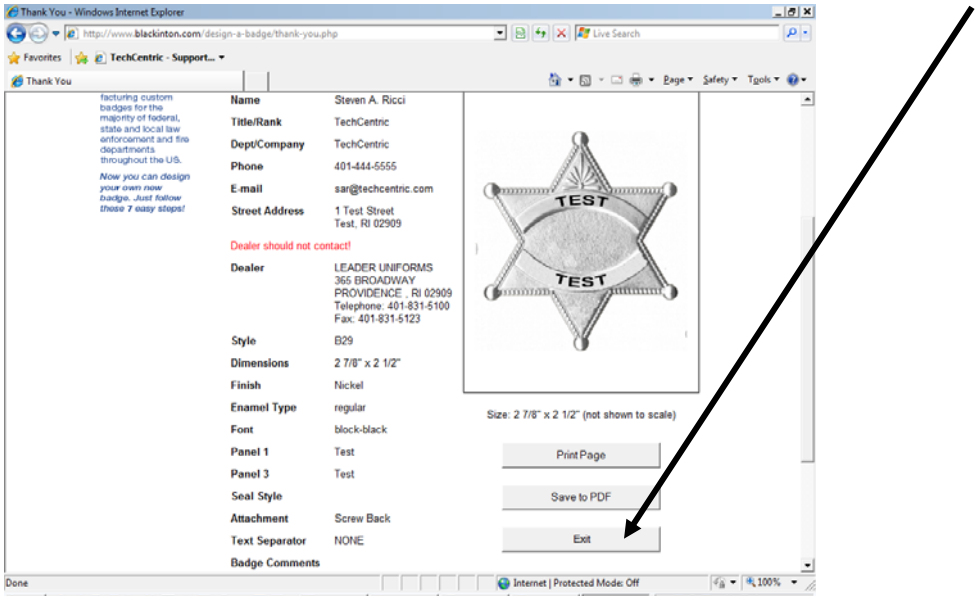
This variable contains the email addresses that will be used to return the data related to the design a web visitor creates using the Design-A-Badge web application. Multiple email addresses must be separated by a semicolon.

This field also has special usage when used in conjunction with feature switches (see below).

_buttonText:

var _buttonText = 'Exit';

This variable contains the value that will be placed on the **BUTTON FACE**.



_userContactDetails:

var userContactDetails =
'skipcontact=no&full_name=.&title_rank=.&department_company=.&phone=000 000 0000&email=user@domain.com&street_address=.&';

To disable or enable the capturing of contact information, set the value highlighted above to 'yes' or 'no'. When set to 'no', the values will default to those indicated above and the "contact information" web page will be suppressed.

dealerURL:

var dealerURL = getClientURL()+'postback.asp';

var dealerURL = "www.dealersite.com/dab/postback.asp';

This variable contains the filename and path of the required server side code (see section *server side code* below). This file will be utilized to direct the Design-A-Badge Extensions where to send the design data using an HTTP POST, which occurs when the website visitor clicks the "EXIT" button at the end of the Design-A-Badge process.

The posting process can be configured to occur immediately when the confirmation page ("thankyou.php") renders (_postOnLoad = true). This approach would be used in a non-shopping cart environment where the button face is NOT changed and the user simply clicks the EXIT button or closes the web browser.

The only modifications that would be made to this variable would be for "server side technology", "path" or "page name".

For Classic ASP, the filename should be "postback.asp". For ASP.NET, the filename should be "postback.aspx". For PHP, the filename should be "postback.php". The name of the file could be changed to something else if needed, but the name of the file and the value of this variable MUST match and the location of the file used to receive the post-back must be on an accessible area of the dealer website. Additionally, the specified file must be able to receive an HTTP-POST.

The location the redirect server file is placed on the dealer site must be reflected within the ***dealerURL***.

The example below would be for a Classic ASP website and would require the placement of the file 'postback.asp' at the same level of the hosting page *(the page that will host the Design-A-Badge <IFRAME>)*. The getClientURL() is a utility function that determines the base URL automatically.

```
var dealerURL = getClientURL()+'postback.asp';
```

The ***dealerURL*** could also contain a full URL without using the utility function.

```
var dealerURL = "www.dealersite.com/dab/postback.asp';
```

<IFRAME> HTML

The coding segment below must be placed within the hosting page along with the code from the "head.js" file (see above). This coding segment should be placed within the <BODY> section of the hosting page in the location you would like the Design-A-Badge web application to render. The screen footprint for the Design-A-Badge web application is approximately 800px.

Copy and paste the code below, do not try to type it visually, as you may make a typographical error.

```
<iframe scrolling='auto' id='dabframe' frameborder='0' width="100%" height="100%">
  <p>This browser does not support the HTML element IFRAME.  Please obtain a browser that is compatible.</p>
</iframe>
```

This code listed above can be found within the file body.htm, which is included in the supporting file download (see overview section above).

Server Side Code

Supporting server side code is contained within the ZIP file which can be downloaded using the link provided within the overview section of this document.

After selecting the appropriate supporting files based on your web hosting platform, the supporting file(s) must be placed within your dealer site in a location that is accessible from the Internet. The supporting file(s) must be able to accept an HTTP POST from the Internet.

The file is offered in three server technologies:

- Classic ASP,
- ASP.NET (C#)
- PHP.

The name of the file, *postback.asp*, *postback.aspx/postback.aspx.cs*, or *postback.php* contains a single line of server side code. The example below is shown using Classic ASP.

The lines of code show below take the Design-A-Badge data that is received via an HTTP-POST and writes it to text file on the dealer's web server and then redirects the web visitor's browser to the URL you provide. This process will occur either when the visitor clicks the "EXIT" button on the last step of the Design-A-Badge process OR when then the page containing the "EXIT" button loads into the browser (see the _postOnLoad variable detailed above).

The setting off the _postOnLoad variable will determine what lines of cost MUST BE CHANGED in the postback.asp, postback.aspx.cs, or postback.php pages. If _postOnLoad=false, then the final line of code in the postback page would be a call to instruct the server to send an HTTP REDIRECT header back to the client browser, which will send the web visitor to the page indicated. The call is already written in the supporting file for each supported server technology. All that is needed is to uncomment the code line.

If the _postOnLoad=true, then the final line of code will be an instruction to send the URL back to the visiting web browser, which will cause redirection. It is a different line of code that must be uncommented.

Both lines of code SHOULD not be used at the same time.

In coding sample below, which uses Classic ASP, the file containing the data posted will be saved to a file, which is generated within a sub directory at the same directory tree level as the hosting page. Once saved the web page is redirection to ***www.google.com***. This is an example of a configuration where _postOnLoad would be set to 'false'.

Changing the parameter within the "Server.MapPath" call would enable the text file to be saved in a different location, where it could be processed by other logic within the dealer website.

It is also important to note that this page does NOT have to be used at all and the HTTP POST could be directed to another page within the dealer site, like a page that adds items to a shopping cart. See the ***dealerURL*** variable described above.

Two fields are posted to the page specified within the ***dealerURL*** variable (see above).

dabID

This value is a unique value that is related to the design transaction. I will only be used once, and is never duplicated for anyone.

Example: *vhbdab1256564524452*

dabData

This value contains all of the information related to the design process, including live links to the IMAGE rendering (including the custom design badge image the web visitor created), and XML representation of the data as well as the raw design data. This field is a JSON (JavaScript Object Notation) format. The best way to process this field is using a JSON utility class or the .NET framework.

There are utility classes available to handle JSON server-side for .NET, PHP and Classic ASP programming technologies.

Example:

{'Name':'StevenA.Ricci','Title/Rank':'President','Dept/Company':'Staffing','Phone':'401-444-5555','E-mail':'sricci70@msn.com','StreetAddress':'2SmithStreetProvidence,RI02909','Customercontact':'Dealershouldnotcontact!','Dealer':'MARKNORMAND','dealeraddr1':'178BARNEYST','dealeraddr2':'RUMFORD,RI02916','dealerphone':'mnormand@blackinton.com','Style':'B29','Dimensions':'27/8"x21/2"(imagenottoscale)','Finish':'Hi-Gloreg;','EnamelType':'hard','Font':'block-blue','Panel1':'Captin','Panel3':'Police','SealStyle':'arkansas','Attachment':'SafetyCatch','TextSeparator':'NONE','BadgeComments':'Plesespaceletters','ContactComments':'NoComment','badgeURL':'http://localhost/blackinton_com/design-a-badge/data/pdf/vhbdab1256573878662.png','badgePDF':'http://localhost/blackinton_com/design-a-badge/data/pdf/vhbdab1256573878662.pdf','RESTXML':'http://localhost/blackinton_com/design-a-badge/generate-xml.php?u = vhbdab1256573878662', 'Base Price':'54.50','Enamel':'0.00','Struck Solid':'0.00','Seal':'0.00','Seal Style':'','':''}

```
<%
Option Explicit

Dim strFileName
Dim objItem

Dim objFileSystem ' fileSystemObject
Dim objTextObject ' textStreamObject

strFileName = Server.MapPath("postdata") & "\" & Request.form("dabID") & ".completed"
```

```
Set objFileSystem = Server.CreateObject("Scripting.FileSystemObject")
Set objTextObject = objFileSystem.OpenTextFile(strFileName, 2, true)

For Each objItem In Request.form
        objTextObject.write objItem & " = " & Request.form(objItem) & VbCrLf
Next

objTextObject.close
Set objTextObject = Nothing
Set objFileSystem = Nothing

'-------------------------------------------------------------------------------------------------------------
' if _postOnLoad=true is set within the client-side supporting code, then ' Response.Write()
' must be uncommented and Response.Redirect must be commented
'-------------------------------------------------------------------------------------------------------------
 response.write("http://www.msn.com")

'-------------------------------------------------------------------------------------------------------------
' if _postOnLoad=false is set within the client-side supporting code, then  Response.Write()
' must be commented out and Response.Redirect must be uncommented
'-------------------------------------------------------------------------------------------------------------
' response.redirect("http://www.msn.com")

%>
```

Additional Support

For additional support regarding this type of integration as well as other integration options, please email support@techcentric.com.