tegrita CONSULTING GROUP

CASL IMPLEMENTATION HANDBOOK

INTRODUCTION

Canada's Anti-Spam Legislation (CASL) is a 2014 legislation that, among other things, regulates when, how and to whom marketing and promotional emails may be sent. CASL generally applies to emails sent from Canada and to emails sent to Canadian residents.

CASL has strict consent requirements. Generally an organization needs express opt-in consent to send commercial emails. There are some limited implied consent exceptions, and CASL as a whole means that businesses need to regulate and manage their subscription lists carefully. CASL is also complex and broadly worded, and imposes significant penalties (including creating personal liability for directors and officers) for non-compliance. We therefore recommend that every organization engaged in email marketing get legal advice from a qualified professional regarding CASL compliance. As described in more detail in the disclaimer below, Tegrita does not and cannot give advice or assurances about whether a particular solution or process complies with CASL.

Web forms are one way to obtain express CASL consent to opt-in to a mailing list. When a user fills out a web form and is asked to give express CASL consent, CASL requires that the initial opt-in confirmation (checkbox, textbox, etc.) presented to the user cannot be prepopulated, so the user must actively check the checkbox to opt-in (or otherwise actively indicate consent).

At Tegrita, one of our core values is Client Enablement, and as such we always strive to give our clients the best solutions to suit their specific needs. With the advent of CASL, we've encountered several usecases requiring specialized solutions.

To that end, we've developed three different web form solutions integrated with Eloqua for our clients that demonstrate some possible approaches to trying to comply with CASL, and the following samples demonstrate the unique functionality of each of these real-life solutions.

For live demonstrations of these approaches, we invite you to review our online implementation here: http://casl.tegrita.com

Proficiency Assumptions: *These instructions are provided with the assumption that the user implementing the solution is familiar with the Eloqua Forms and Landing Page assets, as form integration details are key. In addition, the user should have a basic understanding of Javascript and HTML.*

DISCLAIMER

Canada's anti-spam legislation ("CASL") is still in its infancy. There is little precedent or guidance - and there is therefore uncertainty - as to what business activity the CASL regulators will deem to be non-compliant with CASL. For this reason, the information in this Handbook and our sample web-form solutions are provided for information purposes only. They are not, and should not be taken as, legal advice. You should not rely on, or take or fail to take any action, based upon this information. Your use or reliance on this Handbook, the contents of this Handbook or the information provided in this Handbook will be at your sole risk. Tegrita Corporation and its legal counsel make no representation or warranty of any kind regarding this Handbook, the contents of this Handbook or the information provided through this Handbook. Subject to applicable law, in no event will Tegrita Corporation or its employees, consultants, members, or legal counsel be liable for damages of any kind arising out of your use of or reliance on this Handbook.

OVERVIEW

CASL Implementation Options:

1. "Always Display - Blank" Opt-in
2. "Persistent, Prepopulated" Opt-in
3. "Persistent, One-time" Opt-in

1. "Always Display – Blank"

This is the simplest implementation of a web form CASL consent mechanism. In this case, the opt-in checkbox is always displayed, and is always blank – requiring the user to opt-in each time the user submits the form.

The advantage to this approach is simplicity in design, but repeated requests to opt-in could annoy users, leading them to stop opting in. Or, the user may overlook the opt-in altogether and erase her opted in status inadvertently. Therefore, we suggest using this form in cases where the user will seldom be faced with the option to opt-in, i.e. a new member registration form.

2. "Persistent, Prepopulated"

In this hybrid implementation of a CASL consent web form, the user is always presented with the opt-in checkbox, but – upon entering a valid email address – her email address is looked up in Eloqua and the opt-in box is prepopulated if the user has already opted in, thus saving the effort of having to click the opt-in box. This also serves as a reminder to the user that she is already opted in, and provides a convenient subscription status update should she choose to deselect the opt-in checkbox.

This method is most useful in an accessible and highly visible form, like a member options panel.

3. "Persistent, One-Time"

The previous two options provide the user with multiple opportunities to select their opt-in preference. This third option is semi-permanent and more effective in maintaining a user's status.

In this web form, a lookup is performed on entry of a valid email as in option 2, but in this case if the user has previously opted in, the opt-in option is suppressed (hidden). Non-opted in users will be presented with the opt-in option, and opted in users will be presented with the opt-out option.

IMPLEMENTATION INSTRUCTIONS

1. "Always Display - Blank"

This is the simplest option to implement. Simply add a "CASL Opt-in" checkbox to your Contact fields in Settings > Setup > Fields & Views, with values "Yes" and "No", value for unchecked should be "No".

In your form, add the checkbox field, default unchecked. In the processing steps, the update type should be set to "Always update." Otherwise, if an opted-in user does NOT check the box on a subsequent submission, her contact record will reflect that she is still opted in.

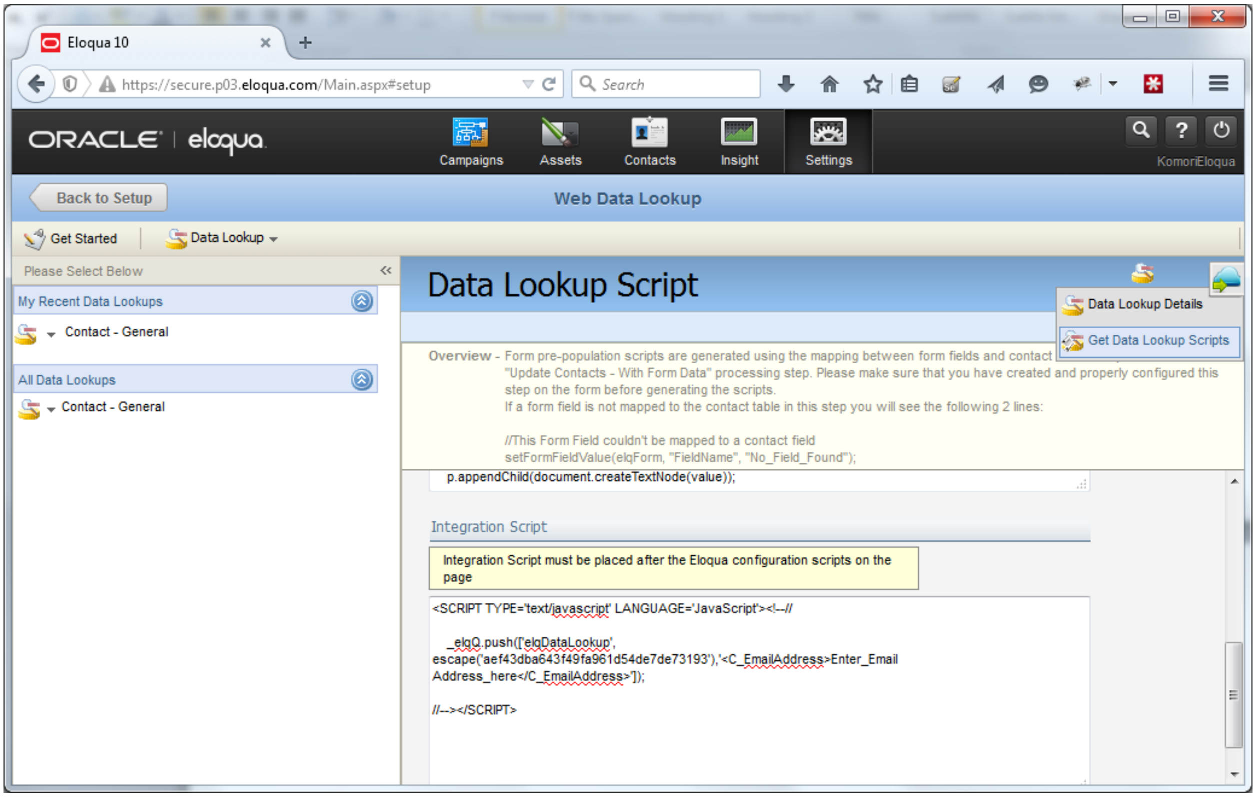
2. "Persistent, Prepopulated" & 3. "Persistent, One-Time"

In the second and third options, a lookup is required on the user's email address.

First, we need to create the lookup in Eloqua. We will need the same Contact field created for option 1 for options 2 and 3.

Web Data Lookups return a collection of fields corresponding to a View. So, we need a CASL View created. Under Settings > Setup > Fields & Views create a new Contact View name CASL, and add the CASL Opt-in Contact field to the new view. Save the view.

Then, under Settings > Setup > Web Data Lookup, create a new "Contacts" data lookup type returning the "CASL" view data (selected under Data Fields Returned). Under "Edit Lookup Criteria", select Email Address as the Data Field and Exact Match as Match Criteria. Save the lookup as "CASL Opt-in By Email".



Data Lookup Script

Overview - Form pre-population scripts are generated using the mapping between form fields and contact "Update Contacts - With Form Data" processing step. Please make sure that you have created and properly configured this step on the form before generating the scripts.
If a form field is not mapped to the contact table in this step you will see the following 2 lines:

```
//This Form Field couldn't be mapped to a contact field
setFormFieldValue(elqForm, "FieldName", "No_Field_Found");
```

```
    p.appendChild(document.createTextNode(value));
```

Integration Script

Integration Script must be placed after the Eloqua configuration scripts on the page

```
<SCRIPT TYPE='text/javascript' LANGUAGE='JavaScript'><!--//

    _elqQ.push(['elqDataLookup',
escape('aef43dba643f49fa961d54de7de73193'),'<C_EmailAddress>Enter_Email
Address_here</C_EmailAddress>']);

//--></SCRIPT>
```

Now in the top right corner of the lookup interface, click on Data Lookup Scripts > Get Data Lookup Scripts and copy the Integration Script for later use.

Back on the landing page HTML, before the closing head tag in the header of your landing page, add the following Eloqua lookup script – remembering to replace your site ID:

```
<script>
   var _elqQ = _elqQ || [];
   var siteID = '[ENTER SITE ID HERE]';
   _elqQ.push(['elqSetSiteId',siteID);
   _elqQ.push(['elqTrackPageView']);
   (function () {
   function async_load() {
   var s = document.createElement('script'); s.type = 'text/javascript'; s.async = true;
   s.src = '//img03.en25.com/i/elqCfg.min.js';
   var x = document.getElementsByTagName('script')[0]; x.parentNode.insertBefore(s, x);
   }
   if (window.addEventListener) window.addEventListener('DOMContentLoaded', async_load, false);
   else if (window.attachEvent) window.attachEvent('onload', async_load);
})();
</script>
```

Next, we'll need to perform a contact lookup the moment the user enters a valid email address. To accomplish this we need to add a few helper functions, including the jQuery library:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
       function validateEmail(email) {
              var re = /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
              return re.test(email);
        }
</script>
```

Now we can use jQuery and the valid email format tester function to perform a lookup the moment a valid email address has been entered/autocompleted.

In the following snippet, we are assuming that there is an input field with id="emailAddress" in the form. Use the elq.push… function call we access in the Web Data Lookups: Integration Script described earlier.

```
<script>
    $(function(){//jQuery wrapper
        $('#emailAddress').on('keyup autocomplete', function(){
            if(validateEmail($(this).val())) _elqQ.push(['elqDataLookup',
escape('[lookupidfromdatalookupscripts]'),'<C_EmailAddress>Enter_Email
Address_here</C_EmailAddress>']);
        });
    });
</script>
```

And finally, we need to define the callback function that will fire if the lookup is successful. Eloqua Web Data Lookups will attempt to call a function called SetElqContent, so let's define the callback function with the actions relevant to options 2 and 3.

For the purposes of Option 2, add id="casl_checkbox" to the opt-in checkbox.

For the purposes of Option 3, add a container DIV around the CASL checkbox and text, and give the container id="casl_section".

```
<script>
    function SetElqContent(){


        // if "CASL Opt-in" is 'Yes'
        if(GetElqContentPersonalizationValue('C_CASLOptIn1') == 'Yes'){


            //OPTION 2: Prepopulate Checkbox
            $('#casl_checkbox').attr('checked', 'checked');


            //OPTION 3: Remove CASL section
            $('#casl_section').remove();


        }
    }
</script>
```