ORACLE

http://docs.oracle.com

Oracle Eloqua Asynchronous Visitor Tracking Scripts

User Guide

24-Aug-2018

Contents

1 Oracle Eloqua asynchronous tracking scripts

The Oracle Eloqua asynchronous tracking scripts allow you to track visits to your website seamlessly without affecting the page load time for your visitors.

When you visit a website with the Oracle Eloqua asynchronous tracking scripts deployed, cookies are placed in your browser. Cookies help identify you as a website visitor according to your specific browser and computer combination in the event that you return to this domain. As visitors browse your website, Oracle Eloqua uses cookies and the visitor's IP address to build a visitor record. Oracle Eloqua cookies remains in the browser until the visitor deletes them or for up to 2 years.

First vs third-party cookies

Oracle Eloqua supports two types of cookie based tracking: third-party and first-party. The difference between third-party cookies and first-party cookies lies in the owner of the cookie.

A third-party cookie is a cookie that has a domain that is different than the domain being visited. A first-party cookie is a cookie that has a domain that is the same as the domain being visited. For example, if you visited example.com, and you saw a cookie for eloqua.com on your browser, this is a third-party cookie because the cookie does not belong to the example.com domain.

Popularity of third-party cookies is declining because they are:

- Blocked by anti-spyware applications
- Blocked by browsers and default privacy settings

First-party cookies are less problematic to use as they are blocked less often. It is difficult to browse the internet without accepting first-party cookies as these cookies identify you as a returning user to the site and help personalize your browsing experience. However, it is not possible to use a first-party cookie to track a single visitor's digital body language across multiple domains owned by your organization.

Note: To enable first-party cookie support, please log in to My Oracle Support (https://support.oracle.com) and create a service request. Review this knowledge base article for more information on what to include in your request.

Executing tracking scripts

In order to ensure the highest probability that Oracle Eloqua can track the page before the visitor leaves it, the tracking scripts are executed after the entire Document Object Model (DOM) is loaded, but before all content is loaded in all browsers (except IE8 and below). The DOM specifies how the objects in a web page are represented, what attributes are associated with each object, and how the objects and attributes can be manipulated. While the DOM is a standard, different browsers support different levels of the DOM and JavaScript standards. For IE8 and below, the entire page must be loaded before requests to the Oracle Eloqua servers are made, they do not support the event that indicates that the DOM has been completely loaded.

Browser support

Tracking scripts are supported and tested in all major browsers, including:

- Internet Explorer 6+
- Firefox 1.5+
- Opera 9+
- Safari 3+
- Chrome

2 Basic page view tracking with third-party cookies

When you visit a website where Oracle Eloqua asynchronous tracking scripts deployed, cookies are placed in your browser. Cookies help websites remember information about your visit and help personalize your next visit.

In a third-party cookie implementation, the domain of the cookie placed in the browser is different than the domain being visited. For example, if you visit example.com and the domain of the cookie placed on your computer is eloqua.com, then this is a third-party cookie. If it the domain of the cookie is example.com, this is a first-party cookie.

Using third-party cookies can be problematic. Third-party cookies are often:

- Blocked by anti-spyware applications
- Blocked by browsers and default privacy settings

2.1 Code sample: Implementing third-party cookies

The following code sample illustrates how to implement third-party cookies on your website.

```
<script type="text/javascript">
    var _elqQ = _elqQ || [];
    _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 =
```

```
'//img.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>
```

When implementing the code sample, note the following:

- `_elqQ` is a queue of commands to push to Oracle Eloqua servers. After the scripts have been loaded into the DOM, the commands are queued up and executed in order.
- The script can be placed anywhere on the page. Since the scripts are executed asynchronously, it does not matter where they are placed on the page. However, if you plan to use other features like data look ups or strict mode, `_elqQ` must be defined before it is used.

3 Basic page view tracking with first-party cookies

Note: To enable first-party cookie support, please log in to My Oracle Support (https://support.oracle.com) and create a service request. Review this knowledge base article for more information on what to include in your request.

When you visit a website where Oracle Eloqua asynchronous tracking scripts deployed, cookies are placed in your browser. Cookies help websites remember information about your visit and help personalize your next visit.

In a first-party cookie implementation, the domain of the cookie placed in the browser is the same as the domain being visited. For example, if you visit example.com and the domain of the cookie placed on your computer is example.com, then this is a first-party cookie. If it the domain of the cookie is different than example.com, this is a third-party cookie.

Implementing tracking using first-party cookies may allow you to track visitors more successfully than using third-party cookies.

- They are blocked less often.
- Many anti-spyware applications and privacy settings do not target first-party cookies.

Good to know

- First-party cookies cannot be shared across domains. So it is not possible to use a first-party cookie to track a single visitor's digital body language across multiple domains owned by your

organization.

- Work with your IT team to ensure your tracking domain has a CNAME record pointing towards Oracle Eloqua in the following syntax: `s[SiteID].hs.eloqua.com`.
- Strict mode tracking using first-party cookies is supported, but strict mode opt-in preferences cannot be set using the form processing step *Web Tracking - Opt-in/Opt-out*.

3.1 Code sample: Implementing first-party cookies

The following code sample illustrates how to implement first-party cookies.

```
<script type="text/javascript">
    var _elqQ = _elqQ || [];
    _elqQ.push(['elqSetSiteId', 'siteId']);
    _elqQ.push(['elqUseFirstPartyCookie',
'<tracking.example.com>']);
    _elqQ.push(['elqTrackPageView']);

    (function() {
        function async_load() {
            var s = document.createElement('script'); s.type
= 'text/javascript';
            s.async = true;
            s.src = '//img.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>
```

When implementing the code sample, note the following:

- Replace *<tracking.example.com>* with your first-party cookie domain in your configuration.
- Replace *siteID* with the three-digit numeric code assigned to your Oracle Eloqua instance. Learn more about obtaining your company's site ID.
- *example.com* should have the same root domain on which your webpages are hosted.
- `_elqQ` is a queue of commands to push to Oracle Eloqua servers. After the scripts have been loaded into the DOM, the commands are queued up and executed in order.
- The script can be placed anywhere on the page. Since the scripts are executed asynchronously, it does not matter where they are placed on the page. However, if you plan to use other features like data look ups or strict mode, `_elqQ` must be defined before it is used.

3.2 Code sample: Retrieving the visitor GUID using a first-party cookie

The following code snippet returns the visitor GUID:

```
function elqGetGuidCookieValue() {
    var name, value, index, cookies=document.cookie.split
(';');
    for (var i = 0; i < cookies.length; i++) {
        index = cookies[i].indexOf('=');
        if (index > 0 && cookies[i].length > index + 1) {
            name = cookies[i].substr(0, index).trim();
            if (name == 'ELOQUA') {
                value = cookies[i].substr(index + 1);
                var subCookies = value.split("&");
                for (var l = 0; l < subCookies.length; l++)
{
                   var subCookie = subCookies[l].split("=");
                   if (subCookie.length == 2 && subCookie[0]
== 'GUID') {
                        return subCookie[1];
                   }
                }
```

```
                }
            }
        }
        return '';
}
```

4 Secure site tracking with first-party cookies

Note: To enable first-party cookie support, please log in to My Oracle Support (https://support.oracle.com) and create a service request. Review this knowledge base article for more information on what to include in your request.

A secure website is a website that's encrypted with Secure Socket Layer (SSL). Additional configuration is required to deploy first-party cookies over secure sites. The tracking domain needs to be configured as a secure microsite in Oracle Eloqua and must have an SSL certificate associated to it.

If this tracking domain is used in the tracking scripts, when a first-party cookie is dropped, the browser will not display any warning messages regarding mixed content and successfully drop a first-party cookie.

After My Oracle Support has enabled first-party cookie tracking for your site and you've set up your secure microsite, tracking scripts are implemented the same as documented in Basic page view tracking with first-party cookies

5 Tracking to multiple Oracle Eloqua instances using JavaScript

Oracle Eloqua's asynchronous tracking scripts allow you to track to more than one instance of Oracle Eloqua if you have a business need for doing so. Configuring that is as simple as calling `elqTrackPageView` twice after setting the different siteIDs. Here's an example:

```
<script type="text/javascript">
    var _elqQ = _elqQ || [];
    _elqQ.push(['elqSetSiteId', '123']);
    _elqQ.push(['elqTrackPageView']);
    _elqQ.push(['elqSetSiteId', '456']);
    _elqQ.push(['elqTrackPageView']);

    (function() {
        function async_load() {
            var s = document.createElement('script');
            s.type = 'text/javascript';
            s.async = true;
            s.src = '//img.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>
```

The highlighted lines show the two extra commands needed to make the extra call to the Oracle Eloqua servers to track a second instance.

6 Tracking custom URLs and referrers using JavaScript

By default, Oracle Eloqua's asynchronous tracking scripts use the current request URL and referrer that the browser passes along to the Oracle Eloqua servers. This works well in the majority of cases, but there are times you may want to customize the URL and/or referrer being passed to Oracle Eloqua's servers to get more value from the system. One classic example is if you have a site that dynamically generates unique URLs that you want to only track as one resource for reporting purposes. To override the URL, simply call the `elqTrackPageView` function with a single parameter, the URL you want to track. For example:

```
<script type="text/javascript">
    var _elqQ = _elqQ || [];
    _elqQ.push(['elqSetSiteId', '123']);
    _elqQ.push(['elqTrackPageView',
'http://www.example.com/downloads/']);

    (function() {
        function async_load() {
            var s = document.createElement('script');
            s.type = 'text/javascript';
            s.async = true;
            s.src = '//img.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>
```

If you would also like to override the referrer as well, adding a second parameter to elqTrackPageView() allows you to do this:

```
<script type="text/javascript">
    var _elqQ = _elqQ || [];
    _elqQ.push(['elqSetSiteId', '123']);
    _elqQ.push(['elqTrackPageView',
'http://www.example.com/downloads/',
'http://www.example.com']);

    (function() {
        function async_load() {
            var s = document.createElement('script');
            s.type = 'text/javascript';
            s.async = true;
            s.src = '//img.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>
```

7 Oracle Eloqua tracking parameters

Oracle Eloqua allows you to create hyperlinks, and then add those hyperlinks to text or images within an asset. Hyperlink parameters allow you to track recipient behavior, attribute it appropriately, and then present desired content to each recipient.

The following parameters are available:

- ElqTrackId
- elqTrack=true
- AssetType and AssetId
- RecipientId
- CampaignId
- SiteId

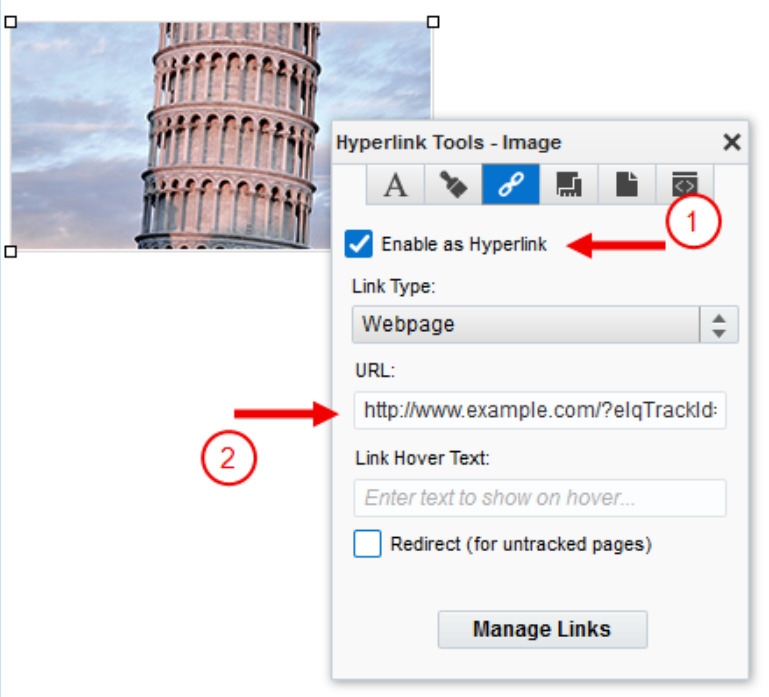
7.0.1 ElqTrackId

The *ElqTrackId* is a unique string of alphanumeric characters that allows Oracle Eloqua to individually identify each linked item in your asset. When you add a hyperlink to an image or text, Oracle Eloqua automatically appends the *elqTrackId* parameter to the link. If you have two images in an email, and both those images link to *http://www.example.com*, then the *elqTrackId* parameter will be different for each of those images.

This parameter also contributes to reporting (such as the visual clickthrough report), and can be useful during debugging.

Example: Oracle Eloqua will convert *http://www.example.com* to:

http://www.example.com/?elqTrackId=9A3A8A5B29B5C3C7C075CE12B85BB8FE



The *elqTrackId* parameter is added to all links in all assets, with a few exceptions:

- *New Email Message* links types do not receive this parameter.
- *System Action* links types do not receive this parameter.

Important: The *View Online Version* link type will receive this parameter when it is added to an email asset.

- Non-HTTP or HTTPS links, such as FTP links, do not receive this parameter.

Important: If you type a link into the URL field that leads to a HTTP site, but you skip typing out the HTTP/HTTPs portion, (and simply begin your URL with *www*), Oracle Eloqua will still append the *elqTrackId* parameter.

Once a parameter is appended to a link, Oracle Eloqua does not revise or remove it. This means that if an asset is generated from a template (for example, the **Save As** function), then the included elqTrackId will not change. To differentiate between such similar links, Oracle Eloqua will refer to a combination of the *elqTrackId* and the *AssetId* (see below).

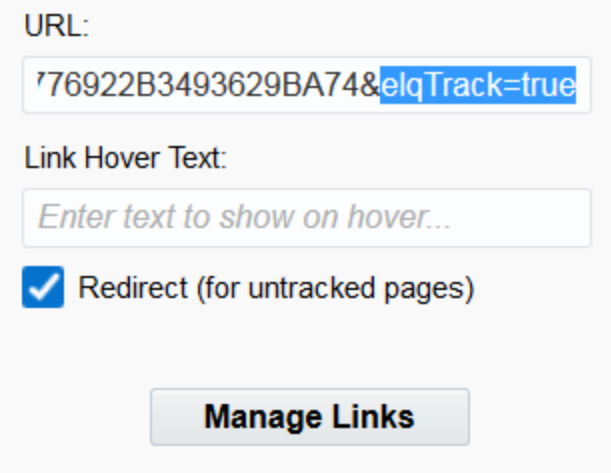
7.0.2 elqTrack=true

The *elqTrack=True* parameter allows Oracle Eloqua to track and report on link activity (such as clickthroughs) for assets on which there are no tracking scripts. It can only be applied to webpage links. The *elqTrack=True*parameter is separate from the *elqTrackId* parameter, although they often appear together on hyperlinks.

Example: Oracle Eloqua will convert *http://www.example.com* to:

http://www.example.com/?elqTrackId=7X1C6B8A6J80L3N6G5T8N7H9E5V5S5A0G1AelqTrack=True

Select the **Redirect (for untracked pages)** check box to add the *elqTrack=True* parameter to your hyperlinks.



When the recipient clicks on the hyperlinked object, the parameter is hidden in the URL in the address bar.

Tip: Selecting the **Redirect** check box will also hide the *elqTrackId* parameter.

7.0.3 AssetType and AssetId

AssetType specifies whether the asset is an email or a landing page. *AssetId* is a unique numerical identifier for each asset that can be used to differentiate between individual assets, assets that were creates via template, and assets that are copies of other assets.

In some cases, when a hyperlinked object is duplicated, the *elqTrackID* parameter will be duplicated as well (see above), and the *AssetId* can help identify links appropriately.

Neither *AssetType* nor *AssetId* are displayed in the user interface, with the exception of the plain-text email editor. Email recipients will see these parameters in plain-text email links, on hover-over display links, and temporarily in the URL address bar when links are being resolved in the browser.

7.0.4 RecipientId

The *RecipientId* parameter is a unique identifier for each email recipient or web page visitor. It can be linked back to a contact or custom object record. When a visitor clicks a link in an email or landing page, the *RecipientId* parameter is passed to the target link, and can then be used to associate clickthroughs and merge content to the appropriate visitor or recipient.

The parameter is present on every link in an email in the format of *elq=<GUID>*. When an email is sent, a Globally Unique Identifier (GUID) is generated and injected into the link for each recipient. Once a recipient has received an email, the *RecipientId* can be found within the URL.

7.0.5 CampaignId

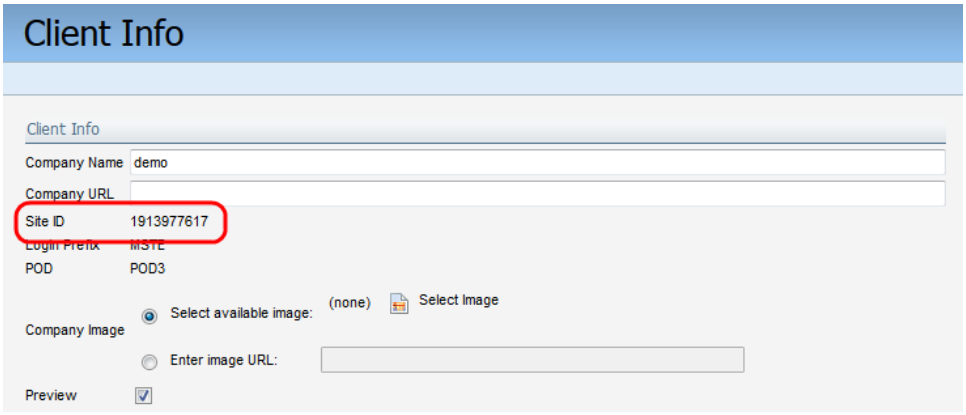
CampaignId is a unique number assigned to an asset when that asset is used as a part of a campaign. If an asset is used in more than one campaign, this number ensures that asset activity is associated with the correct campaign.

See sharing an Oracle Eloqua-hosted form across multiple Oracle Eloqua campaigns for an example of this parameter in use.

7.0.6 SiteId

SiteId identifies your company's assets. The *SiteId* is formatted as *s=<customer side id>*.

If you have administrator privileges, learn your *SiteId* information by clicking **Settings**, then clicking **Company Defaults** in the *Display Preferences* section.



8 Using tracking pixels to track page visits

A tracking pixel is a 1x1-sized pixel image that can be embedded within your landing pages to track page visits. The tracking pixel image is an image tag that points to an endpoint on the Oracle Eloqua servers and accepts the ELQ parameter (unique recipient id). When the image is loaded, the endpoint records the page visit against the recipient ID (which can be resolved to a contact for reporting.) The HTML snippet looks similar to this:

```
<img style="display:none; border: 0" src="http://s
[siteID].t.eloqua.com/visitor/v200/svrGP?pps=3&siteid=
[siteID]&ref=http://google.com/" width="1" height="1"
alt="">
```

In the example above, *[siteID]* where highlighted, represents your site ID.

9 Testing visitor tracking scripts

You can test visitor tracking scripts to ensure that Oracle Eloqua is tracking visits to your tracked pages. The testing utility scans the specified webpage for the following:

- Whether the URL provided is a valid webpage
- Whether Oracle Eloqua can find references to one of the following tracking scripts in the HTML code:
  - `elqcfg.js` and `elqimg.js`
  - `elqcfg.min.js`

> **Note**: Oracle Eloqua is not searching for a JavaScript file, but is searching for the above text that is typically located within the `<head>` tags.

- Whether the cookie is a first-party cookie or third-party cookie
- Validates the tracking script for proper syntax

> **Note**: If one of the tests fails, Oracle Eloqua will not perform the remaining tests.

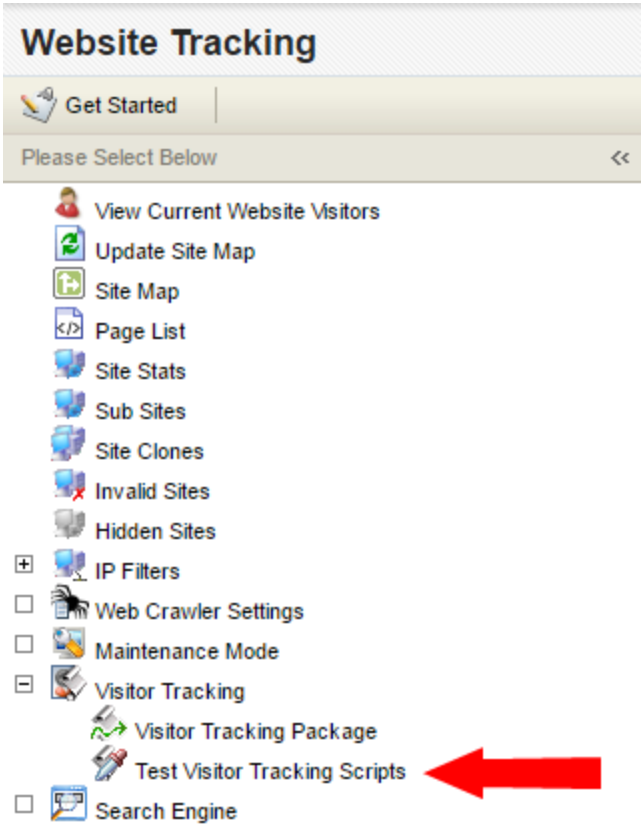
Before you begin:

- Gather the URLs you wish to test. You will need the fully qualified URL, including `https://` and `www`.

- The domains of the URL will need to be added to the domain whitelist.
- Remove any query strings from the URL.

To test visitor tracking scripts:

1. Navigate to **Assets** > **Web Setup**, then click **Tracking**.
2. Expand **Visitor Tracking**, then click **Test Visitor Tracking Scripts**.



3. Enter the URL you wish to track in the blank field, then click **Start Test**.

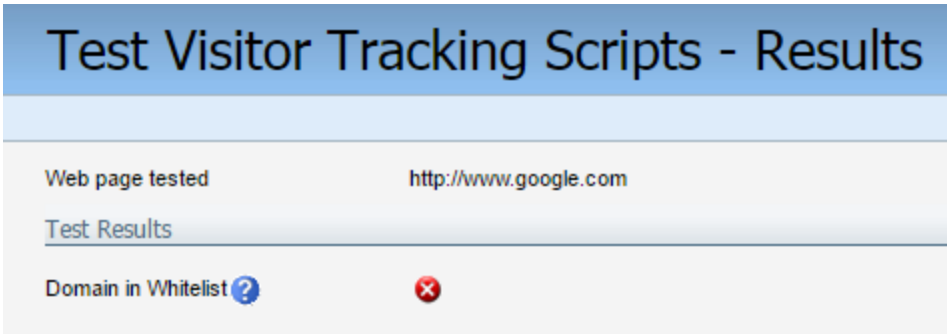
> **Note**: The URL needs to be in the domain whitelist, otherwise Oracle Eloqua returns an error.

Oracle Eloqua then performs the tests and returns the test results.

9.1 Troubleshooting

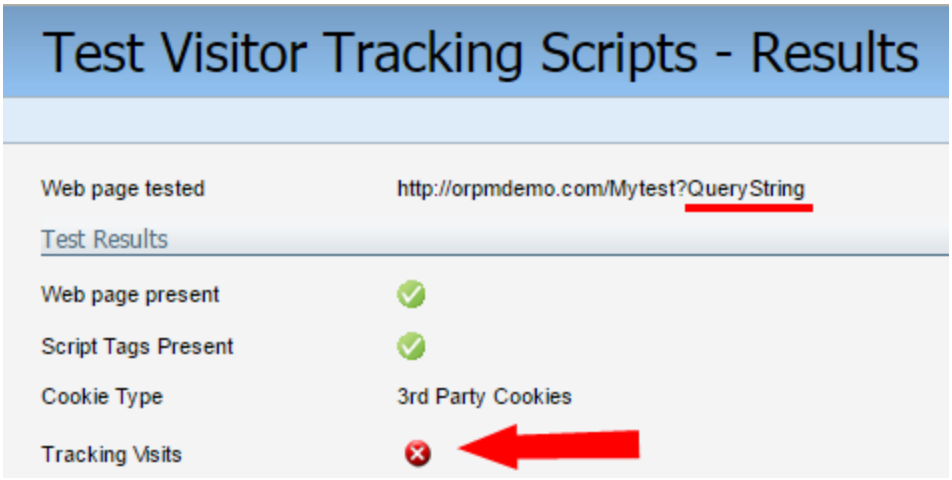
There are a few reasons why Oracle Eloqua might return errors when testing tracking scripts:

- If the **Domain in Whitelist** returns an error, you must add the domain to the domain whitelist.



- If the **Web page present** returns an error, ensure you included all aspects of the URL, and removed any query strings.
- If **Script Tags Present** returns an error, it is possible that the tracking script setup is the issue. If the script was set up using a third-party program, the test may not be able to find the tracking scripts. See this Knowledge Base article for more information.

- If there is an error in **Tracking Visits**:
  - Ensure the site ID for the page you are testing is only comprised of numbers. Non-numeric characters in the site ID will cause an error for **Tracking Visits**.
  - Ensure the URL you inputted does not contain query strings.



  - If the asynchronous tracking script was customized, it is possible Oracle Eloqua can no longer validate that links are being tracked. At this point the **Script Tags Present** test will be a success, but **Tracking Visits** will return an error.

10 Outbound link tracking

Outbound link tracking allows you to track when users click specific links on your site. The links may be to other sites or to a different location within your site. The links could be for web pages or they could point to documents like PDFs or Word documents.

To accomplish outbound link tracking, you will have to modify your current anchor tags by adding an onclick handler. For example, a link that currently looks like this:

```
<a href="http://example.com/documents/whitepaper.pdf"></a>
```

Should be modified to look like this:

```
<a href="http://example.com/documents/whitepaper.pdf"
onclick="_elq.trackOutboundLink(this);return false;"></a>
```

If the above use case isn't valid for your situation, a more generic JavaScript function is exposed for this purpose:

```
_elq.trackEvent
('http://example.com/documents/whitepaper.pdf');
```

Note: The `trackEvent` function above will track that the user has clicked on the particular document or URL, but it won't actually redirect the user. You may need to use this functionality if you have a more complicated workflow, but the first method, using `trackOutboundLink`, is the preferred method.

11 Reposting externally hosted forms

Use these scripts when the form data is reposted to Oracle Eloqua servers from a server-side form processor. In this case it is not possible to get the cookie from the client's browser, so the following scripts should be installed on the form page to send the unique ID along with the form data.

In addition to the regular Oracle Eloqua form elements:

```
 <input type="hidden" name="elqFormName"
value="EloquaFormName" />
 <input type="hidden" name="elqSiteId" value="SiteId" />
```

Add the following two hidden fields to your form:

```
 <input type="hidden" name="elqCustomerGUID" value="">
 <input type="hidden" name="elqCookieWrite" value="0">
```

Then, place the following code anywhere after the main tracking scripts appear on the page:

```
var timerId = null, timeout = 5;

function WaitUntilCustomerGUIDIsRetrieved() {
    if (!!(timerId)) {
        if (timeout == 0) {
            return;
        }
        if(typeof this.GetElqCustomerGUID === 'function') {
            document.forms["EloquaFormName"].elements
["elqCustomerGUID"].value = GetElqCustomerGUID();
            return;
        }
        timeout -= 1;
    }
    timerId = setTimeout("WaitUntilCustomerGUIDIsRetrieved
```

```
()", 500);
    return;
}
window.onload = WaitUntilCustomerGUIDIsRetrieved;
_elqQ.push(['elqGetCustomerGUID']);
```

This code will execute upon the page loading and will wait until the visitor's GUID has been retrieved from the Oracle Eloqua servers and then insert it into the `elqCustomerGUID` form element. Ensure to replace `EloquaFormName` with the actual name of your form as setup in Oracle Eloqua.

12 Flash content tracking

Tracking clicks in Adobe Flash content is easily achievable with the Oracle Eloqua asynchronous tracking scripts. The package you receive from Oracle Eloqua should have a Flash sample in it to help you out. The method used in that sample is outlined below:

Supposing you had a button in your Flash content you want to track, in your click handler, you would add the following code (replacing the parameter values with your own):

ActionScript 2:

```
on (release) {
       // your button click handling code
       elqTrackFlashClick
('http://example.com/content/FlashButton1',
'http://http://example.com/content/referrer');
}
```

ActionScript 3:

```
import flash.external.ExternalInterface;
button_1.addEventListener(MouseEvent.CLICK, fl_
MouseClickHandler);

function fl_MouseClickHandler(event: MouseEvent): void
{
    // your button click handling code
    ExternalInterface.call("elqTrackFlashClick
('http://example.com/content/FlashButton1',
'http://http:/example.com/content/referrer')");
}
```

where "button_1" is the name of your button.

On your page that hosts the flash object, add the `elqTrackFlashClick` JavaScript function in the head section below the standard tracking scripts:

```
<script type='text/javascript'>
       function elqTrackFlashClick(URL, referrer) {
              _elqQ.push(['elqTrackPageView', URL, referrer]);
       }
</script>
```

Putting it all together, your page hosting the Flash content might look like this:

```
<html>

<head>
    <script type="text/javascript">
        var _elqQ = _elqQ || [];
        _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 = '//img.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);
        })();
        function elqTrackFlashClick(URL, referrer) {
            _elqQ.push(['elqTrackPageView', URL, referrer]);
        }
    </script>
</head>
```

```
<body>
    <div id="flashContent">
        <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-
444553540000" width="550" height="400" id="FlashExample"
align="middle">
            <param name="movie" value="FlashExample.swf" />
            <param name="quality" value="high" />
            <param name="bgcolor" value="#ffffff" />
            <param name="play" value="true" />
            <param name="loop" value="true" />
            <param name="wmode" value="window" />
            <param name="scale" value="showall" />
            <param name="menu" value="true" />
            <param name="devicefont" value="false" />
            <param name="salign" value="" />
            <param name="allowScriptAccess"
value="sameDomain" />
            <!--[if !IE]>-->
            <object type="application/x-shockwave-flash"
data="FlashExample.swf" width="550" height="400">
                <param name="movie" value="FlashExample.swf"
/>
                <param name="quality" value="high" />
                <param name="bgcolor" value="#ffffff" />
                <param name="play" value="true" />
                <param name="loop" value="true" />
                <param name="wmode" value="window" />
                <param name="scale" value="showall" />
                <param name="menu" value="true" />
                <param name="devicefont" value="false" />
                <param name="salign" value="" />
                <param name="allowScriptAccess"
value="sameDomain" />
                <!--<![endif]-->
                <a href="http://www.adobe.com/go/getflash">
                    <img
src="http://www.adobe.com/images/shared/
download_buttons/get_flash_player.gif" alt="Get Adobe Flash
player" />
                </a>
                <!--[if !IE]>-->
```

```
            </object>
            <!--<![endif]-->
        </object>
    </div>
</body>
</html>
```

13 Strict mode tracking

> **Note**: To enable strict mode tracking, please log in to My Oracle Support and create a service request. Review this knowledge base article for more information on enabling strict mode tracking.

Oracle Eloqua's asynchronous tracking scripts allow you to implement strict mode tracking. Enabling strict mode allows you to restrict the tracking of visitors to your web page(s). There are three levels in relation to strict mode tracking scripts:

- **Default tracking**: Track all visitors.
- **Require opt-in for visitors by country**: Track all visitors except for those from countries on a restricted country list, unless those visitors opt in for tracking via a permissions request.
- **Require opt-in for all visitors**: Don't track any visitors, unless those visitors opt in for tracking via a permissions request.

Strict mode helps you comply with the European Union's privacy tracking regulations. Member countries of the EU have legislation in place that requires a marketer to receive explicit opt-in consent to track individuals actions online. If you are conducting a campaign that includes EU member states, then it is your responsibility to determine and set which tracking mode permission request (if any) for each of your web pages.

13.1 Code sample: Requiring opt-in for visitors by country

When using the following code, Oracle Eloqua tracks visitors from countries that are not on the Country by IP restriction list, and tracks visitors from restricted countries that have opted-in for tracking:

```
<script type="text/javascript">
   var _elqQ = _elqQ || [];
   _elqQ.push(['elqSetSiteId', 'siteId']);
   _elqQ.push(['elqTrackPageViewOptinByCountry']);
      (function() {
      function async_load() {
         var s = document.createElement('script');
         s.type = 'text/javascript';
         s.async = true;
         s.src = '//img.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>
```

The following code does the same thing as the above snippet, except that it shows a simple default opt-in/out banner at the top of the page to new visitors coming from restricted countries. For visitors who have opted-into tracking, the banner will not be displayed and page views will be tracked. The styling and text of the Opt-in banner can be customized.

```
<script type="text/javascript">
   var _elqQ = _elqQ || [];
   _elqQ.push(['elqSetSiteId', 'siteId']);
   _elqQ.push
(['elqTrackPageViewDisplayOptInBannerByCountry']);
   (function () {
      function async_load() {
         var s = document.createElement('script');
         s.type = 'text/javascript';
         s.async = true; s.src =
'//img.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);
    })();
        function elqVisitorTrackingOptIn() {
            _elqQ.push(['elqOptIn']);
        }
        function elqVisitorTrackingOptOut() {
            _elqQ.push(['elqOptOut']);
        }
</script>
```

Important: If strict mode is enabled but you do not have any countries on your restricted country list, then Oracle Eloqua will not track any visitors to the web page even if there is a call for the require opt-in for visitors by country function. After you add countries to the list, Oracle Eloqua will then restrict tracking to those countries. This also affects the tracking of redirect links from emails and landing pages. Learn more from this knowledge base article on how to add countries to the list.

13.2 Code sample: Requiring opt-in for all visitors

When using the following code, Oracle Eloqua excludes *all* visitors from tracking unless they have explicitly opted into tracking:

```
<script type="text/javascript">
    var _elqQ = _elqQ || [];
    _elqQ.push(['elqSetSiteId', 'siteId']);
    _elqQ.push(['elqTrackPageViewOptinAll']);
    (function () {
```

```
      function async_load() {
         var s = document.createElement('script');
         s.type = 'text/javascript';
         s.async = true; s.src =
'//img.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>
```

The following code does the same thing as the above snippet, except that it shows a simple default opt-in banner at the top of the page to new visitors. For visitors who have already opted-in to tracking, the banner will not be displayed and page views will be tracked. The styling and text of the Opt-in banner can be customized.

```
<script type="text/javascript">
   var _elqQ = _elqQ || [];
   _elqQ.push(['elqSetSiteId', 'siteId']);
   _elqQ.push(['elqTrackPageViewDisplayOptInBannerForAll']);
   (function () {
      function async_load() {
         var s = document.createElement('script');
         s.type = 'text/javascript';
         s.async = true; s.src =
'//img.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);
   })();
```

```
   function elqVisitorTrackingOptIn() {
      _elqQ.push(['elqOptIn']);
   }
   function elqVisitorTrackingOptOut() {
      _elqQ.push(['elqOptOut']);
   }
</script>
```

14 Customizing the Opt-in/out banner

You can create your own banner rather than using the built-in default. For example:

```
<script type="text/javascript">
       function elqCreateOptInBanner() {
               if (navigator.appVersion.indexOf('MSIE') != -1) {
                       var css = '.elqOptInBanner {position: absolute;
 left: 0px; width: 100%; border:solid 1px #c0c0c0;
background-color:#e1e1e1;
 font-size:11px; font-family:verdana; color:#000;
padding:5px;} .elqOptInBannerText
 {float: left; text-align:left; width:96%;} .elqButton
{font-size:11px; color:#000;
 padding:3px;} .elqClose {float:right; font-size:14px; font-
weight:bold;
 cursor:pointer; padding-right:15px;} ';
               }
               else {
                       var css = '.elqOptInBanner {position: fixed; top
 left: 0px; width: 100%; border:solid 1px #c0c0c0;
background-color:#e1e1e1;
 font-size:11px; font-family:verdana; color:#000;
padding:5px;} .elqOptInBannerText
 {float: left; text-align:left; width:96%;} .elqButton
{font-size:11px; color:#000;
 padding:3px;} .elqClose {float:right; font-size:14px; font-
weight:bold; cursor:pointer;
 padding-right:15px;} ';
               }
               var style = document.createElement('STYLE');
               style.type = 'text/css';
               if (style.styleSheet) {
                       style.styleSheet.cssText = css;
               } else {
                       style.appendChild(document.createTextNode(css));
               }
               var head = document.getElementsByTagName('head')[0];
               head.appendChild(style);
               var div = document.createElement('div');
```

```
                div.setAttribute('id', 'elqOptInBannerDiv');
                div.className = 'elqOptInBanner';
                div.innerHTML = '<div class="elqOptInBannerText" >
This website would like to use cookies to store information
on your computer which will
help to customize your interactions with us and provide you
with a better browsing experience.
 You may delete and block cookies from this site, but parts
of the site may not function as a
 result. More information about cookies and your choices can
be found in our Privacy Policy.
 To accept cookies from this site, please click the \'I
Accept\' button below.   
<input type="button" id="elqOptInButton" value="I Accept"
onClick="elqVisitorTrackingOptIn();
document.getElementById(\'elqOptInBannerDiv\').style.display
= \'none\';" class="elqButton" />
  <input type="button" id="elqOptOutButton"
value="I Decline"
onClick="elqVisitorTrackingOptOut();document.getElementById
(\'elqOptInBannerDiv\').style.display
 = \'none\';" class="elqButton" /></div><div
class="elqClose" onclick="document.getElementById
(\'elqOptInBannerDiv\').style.display = \'none\';">x</div>';
                document.body.appendChild(div);
        }
</script>
```

Note: The function name cannot be changed. The tracking scripts code will call **elqCreateOptInBanner**.

15 Opting-in and out

The Oracle Eloqua tracking scripts allow users to opt-out of tracking at the site level as well as globally for all domains. Opting-out at the site level involves creating a new cookie called ELQOPTOUT and setting a value to indicate that for this particular site, the visitor should not be tracked in the Oracle Eloqua system. The user's GUID is not removed and so if they later choose to opt back in, they will resume tracking against the same visitor record. Opting-out at the global level removes their GUID from the cookie and thus makes them invisible to our servers. If they opt back in, they will get a new GUID and will be tracked as a new visitor.

> **Note**: The opt-in/opt-outs for first and third-party cookie tracking are independent of each other. For example, if a site is using first-party cookie tracking and a site visitor has both a first-party cookie opt-in and a third-party cookie opt-out, only the first-party cookie opt-in will be used and the visitor will be tracked. This is an edge case that is relevant primarily when you are transitioning a domain from third-party to first-party cookie tracking as part of the beta program and a single visitor has both a third and a first-party cookie.

The following code snippet returns visitor opt-in status for first-party cookie tracking. It returns **1** if the visitor is opted in and **0** if the visitor is opted out. If the visitor is neither opted in nor opted out, an empty string is returned.

```
function elqGetOptInCookieValue() {
      var name, index, cookies = document.cookie.split(';');
      for (var i = 0; i < cookies.length; i++) {
            index = cookies[i].indexOf('=');
```

```
                if (index > 0 && cookies[i].length > index + 1) {
                        name = cookies[i].substr(0, index).trim();
                        if (name == 'OPTIN') {
                                return cookies[i].substr(index + 1);
                        }
                }
        }
        return '';
}
```

16 Cookie laws country list

Member countries of the EU have legislation in place that requires a marketer to receive explicit opt-in consent to track individuals actions online. If you are conducting a campaign that includes EU member states, then it is your responsibility to determine and set which tracking mode permission request (if any) for each of your web pages.

Below is an example list of countries that have had cookies laws implemented:

- Austria
- Belgium
- Bulgaria
- Cyprus
- Czech Republic (opt-out)
- Denmark
- Estonia
- Finland
- France
- Hungary
- Ireland
- Italy
- Latvia
- Lithuania
- Luxembourg
- Malta

- Netherlands
- Poland
- Portugal
- Slovakia
- Slovenia
- Spain
- Sweden
- United Kingdom

👉 **Tip**: Enabling Eloqua's strict mode tracking can help you comply with the regulations of these countries. Strict mode prevents you from tracking what web pages and landing pages that a customer visits, or tracking email clickthroughs.

17 Using data lookups

Data lookups allow you to retrieve contact, visitor, or custom object data. Implementing a data lookup on your website allows you to personalize web pages with contact or visitor information.

Using web data look ups, you can do the following:

- Display different web pages based on visitor profile data
- Display a specific web page to contacts belonging to an email group
- Allow entry to a portion of your site to known contacts
- Pre-populate a form

To implement a data lookup, you will need the following:

1. In Oracle Eloqua, create a new data lookup. You will need the *Data Lookup Key* call the web data lookup.
2. Ensure Oracle Eloqua tracking scripts are implemented on your web page.

> 👉 **Tip**: To help implement a data lookup, use the sample code generated by Oracle Eloqua. To view the sample code, open the data lookup and click **Data Lookup Options > Get Data Lookup Scripts**. Learn more about creating a data lookup.

Writing contact data to a web page

The following describes writing contact data retrieved from a contact data lookup to a web page.

A `div` is placed on the page and will be populated with the contact details returned by the data lookup.

```
<div id="contactinfo">
  <!-- contact info will be added here -->
</div>
```

A JavaScript function dynamically creates paragraph elements with the retrieved data and places them in the `div` above.

```
function CreateRow(label, value) {
  var p = document.createElement('p');
  var b = document.createElement('b');
  p.appendChild(b);
  b.appendChild(document.createTextNode(label));
  p.appendChild(document.createTextNode(value));
  document.getElementById('contactinfo').appendChild(p);
}
```

A callback function `SetElqContent` is used to return the data from the lookup. The function name is important and must be called `SetElqContent`. The data that is returned depends on how the data lookup was set up. In the following example, contact data is returned.

```
function SetElqContent() {
  CreateRow('Contact Info: ',
GetElqContentPersonalizationValue(''));
  CreateRow('First Name: ',
GetElqContentPersonalizationValue('C_FirstName'));
  CreateRow('Last Name: ', GetElqContentPersonalizationValue
('C_LastName'));
  CreateRow('Email Address:
',GetElqContentPersonalizationValue('C_EmailAddress'));
  CreateRow('Salesperson: ',
GetElqContentPersonalizationValue('C_Salesperson'));
  CreateRow('Company: ', GetElqContentPersonalizationValue
```

```
('C_Company'));
  CreateRow('Address 1: ', GetElqContentPersonalizationValue
('C_Address1'));
  CreateRow('City: ', GetElqContentPersonalizationValue('C_
City'));
  CreateRow('State or Province:
',GetElqContentPersonalizationValue('C_State_Prov'));
  CreateRow('Zip or Postal Code:
',GetElqContentPersonalizationValue('C_Zip_Postal'));
  CreateRow('Business Phone: ',
GetElqContentPersonalizationValue('C_BusPhone'));
  CreateRow('Country: ', GetElqContentPersonalizationValue
('C_Country'));
  CreateRow('Industry: ', GetElqContentPersonalizationValue
('M_Industry1'));
  CreateRow('Revenue: ', GetElqContentPersonalizationValue
('M_Revenue1'));
  CreateRow('Website: ', GetElqContentPersonalizationValue
('M_WebSite1'));
  CreateRow('Date Created:
',GetElqContentPersonalizationValue('M_DateCreated'));
}
```

The `_elqQ.push()` is called that requests the data lookup from the Oracle Eloqua servers.

```
_elqQ.push(['elqDataLookup', escape('<Data_Lookup_Key>'),
'<Data_Lookup_Criteria>']);
```

`Data_Lookup_Key` is a global unique identifier (GUID) that is generated when you create the data lookup. You can retrieve it by opening up the web data lookup in Oracle Eloqua. The `Data_Lookup_Criteria` (if required) is the criteria you set up to identify the contact or visitor. For example, email address `<C_EmailAddress>user@example.com</C_EmailAddress>`. Learn more about creating a data lookup.

Trying it all together, we get the following:

```
<html>
<head>
    <script type="text/javascript">
        var _elqQ = _elqQ || [];
        _elqQ.push(['elqSetSiteId', '123']);
        _elqQ.push(['elqTrackPageView']);

        (function () {
            function async_load() {
                var s = document.createElement('script');
s.type = 'text/javascript'; s.async = true;
                s.src = '//img.en25.com/i/elqCfg.min.js';
                var x = document.getElementsByTagName
('script')[0]; x.parentNode.insertBefore(s, x);
            }
            if (window.attachEvent) { window.attachEvent
('onload', async_load); }
            else { window.addEventListener
('DOMContentLoaded', async_load, false); }
        })();
    </script>
    <script type="text/javascript">
        function CreateRow(label, value) {
            var p = document.createElement('p');
            var b = document.createElement('b');
            var label = document.createTextNode(label);
            var value = document.createTextNode(value);
            p.appendChild(b);
            b.appendChild(label);
            p.appendChild(value);

            document.getElementById
('contactinfo').appendChild(p);
        }

        function SetElqContent() {
            CreateRow('Contact Info: ',
GetElqContentPersonalizationValue(''));
            CreateRow('First Name: ',
GetElqContentPersonalizationValue('C_FirstName'));
            CreateRow('Last Name: ',
GetElqContentPersonalizationValue('C_LastName'));
```

```
            CreateRow('Email Address: ',
GetElqContentPersonalizationValue('C_EmailAddress'));
            CreateRow('Salesperson: ',
GetElqContentPersonalizationValue('C_Salesperson'));
            CreateRow('Company: ',
GetElqContentPersonalizationValue('C_Company'));
            CreateRow('Address 1: ',
GetElqContentPersonalizationValue('C_Address1'));
            CreateRow('City: ',
GetElqContentPersonalizationValue('C_City'));
            CreateRow('State or Province: ',
GetElqContentPersonalizationValue('C_State_Prov'));
            CreateRow('Zip or Postal Code: ',
GetElqContentPersonalizationValue('C_Zip_Postal'));
            CreateRow('Business Phone: ',
GetElqContentPersonalizationValue('C_BusPhone'));
            CreateRow('Country: ',
GetElqContentPersonalizationValue('C_Country'));
            CreateRow('Industry: ',
GetElqContentPersonalizationValue('M_Industry1'));
            CreateRow('Revenue: ',
GetElqContentPersonalizationValue('M_Revenue1'));
            CreateRow('Website: ',
GetElqContentPersonalizationValue('M_WebSite1'));
            CreateRow('Date Created: ',
GetElqContentPersonalizationValue('M_DateCreated'));
        }

        _elqQ.push(['elqDataLookup', escape
('434C5E6250C04FCBB0D4FC5413F9A40A'), '
<C_EmailAddress>user@example.com</C_EmailAddress>']);
    </script>
</head>
<body>
    <div id="contactinfo">
    </div>
</body>
</html>
```

17.1 Creating web data lookups

Data lookups allow you to retrieve contact, visitor, or custom object data. Implementing a data lookup on your website allows you to personalize web pages with contact or visitor information.

Using web data look ups, you can do the following:

- Display different web pages based on visitor profile data
- Display a specific web page to contacts belonging to an email group
- Allow entry to a portion of your site to known contacts
- Pre-populate a form

You can use a web data lookup to retrieve the following type of information:

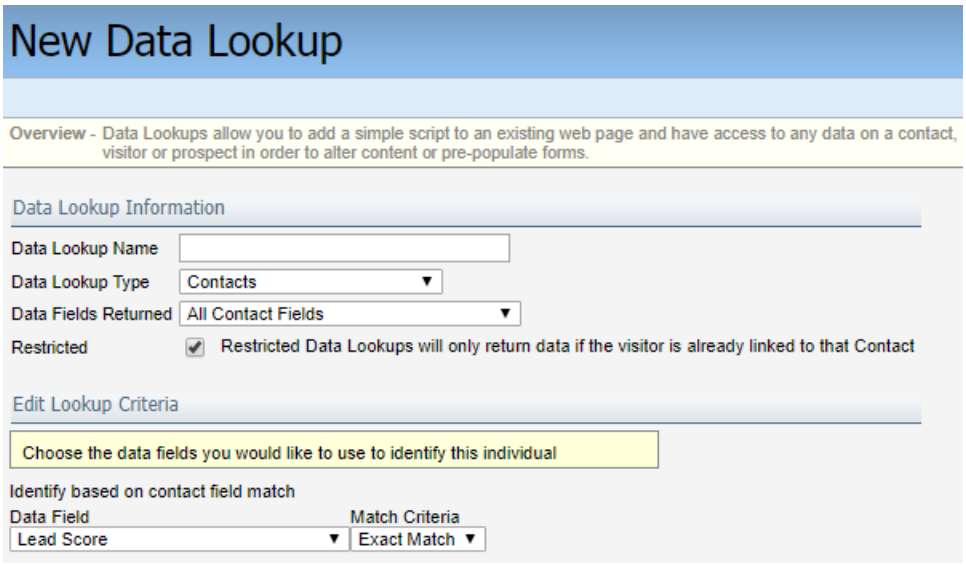
- Contact fields
- Visitor profile fields
- A count of custom data records for a given custom object
- Custom data record fields
- Email group membership

Note: Contact data is limited to those contact fields enabled for web data lookups.

To create a web data lookup:

1. Navigate to **Assets** > **Website Setup** and then select **Web Data Lookup**.

2. Click **Data Lookup** > **New Data Lookup**.



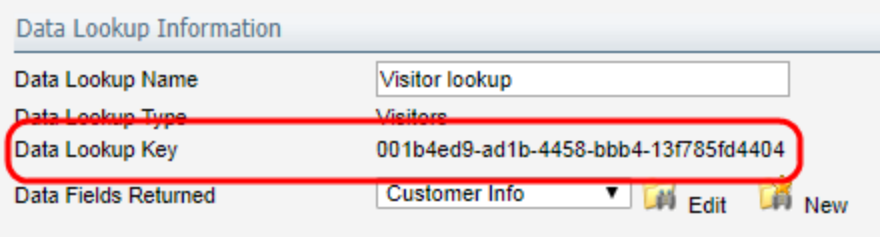
3. Give the data lookup a name and choose the type of data lookup that you want to create. You can choose from:

   - **Contacts**: Used to return contact field data. Contact data is retrieved based on the contact field you specify to match the contact on. If you are creating a contact data lookup, ensure the **Restricted** option is selected. This option prevents returning contact data if the visitor profile is not linked to the contact.
   - **Visitors**: Used to return visitor profile fields. Visitor data is always retrieved using the Oracle Eloqua cookie global unique identifier (GUID).
   - **Custom Object Records**: Used to return custom data record. Custom data records are retrieved based on the field you specify to match the contact on.
   - **Custom Objects**: Used to return the number of custom data records in a given custom object. Custom objects are retrieved using the custom object ID.
   - **Contact Group Memberships**: Used to return a contact's email group membership. Group membership data is retrieved based on the contact field you specify to match the contact on.

4. Choose an option:
    - If you are creating a contact and visitor data lookups, choose the view from the **Data Fields Returned** drop-down list. The data lookup only returns fields that are in the view.
    - If you are creating a custom object lookups, choose the custom object. This determines which custom object to use in the web data lookup.
5. If applicable, choose the field that you want to use to identify the contact in the **Edit Lookup Criteria** area. For example, contact email address. Contacts are matched to an Oracle Eloqua contact record based on this field.

    After you save the data lookup, you can add additional match criteria. This can help ensure that correct contact record is returned by the data lookup.
6. Save the web data lookup.

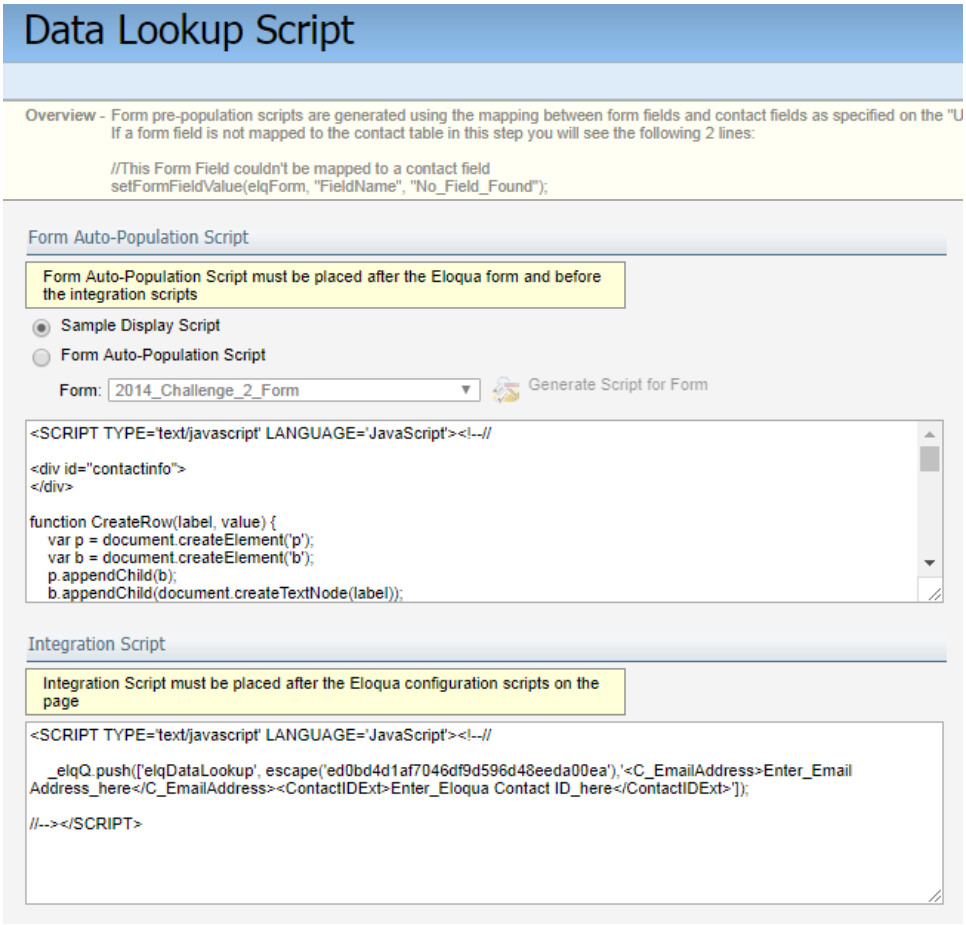
After you save the data lookup, Oracle Eloqua generates a unique key for the lookup. You need this key to implement to the data lookup on your website.



Data lookup sample code

After you save the data lookup, Oracle Eloqua generates JavaScript sample code that you can use to implement the data lookup on your website.

To view the sample code, click **Data Lookup Options > Get Data Lookup Scripts**.



The following sample code is provided:

- Sample display script: This sample shows you how to return the data from the data lookup. If you created a contact or visitor data lookup, this sample includes references to the fields returned by the lookup.
- Form auto-population script: This samples shows you to pre-populate an Oracle Eloqua from with data returned from the lookup. To use this, select the from and then click *Generate Script for Form*.

> **Tip**: If you want to pre-populate an Oracle Eloqua form, it is recommend that you use field merges to populate the form field.

- Integration: This sample shows you how to call the data lookup from your web page.

When you implement the data lookup on your website, the web page must also have the Oracle Eloqua tracking scripts implemented. Learn more about using a data lookup on a web page or data lookup examples.

17.2 Examples of data lookup usage

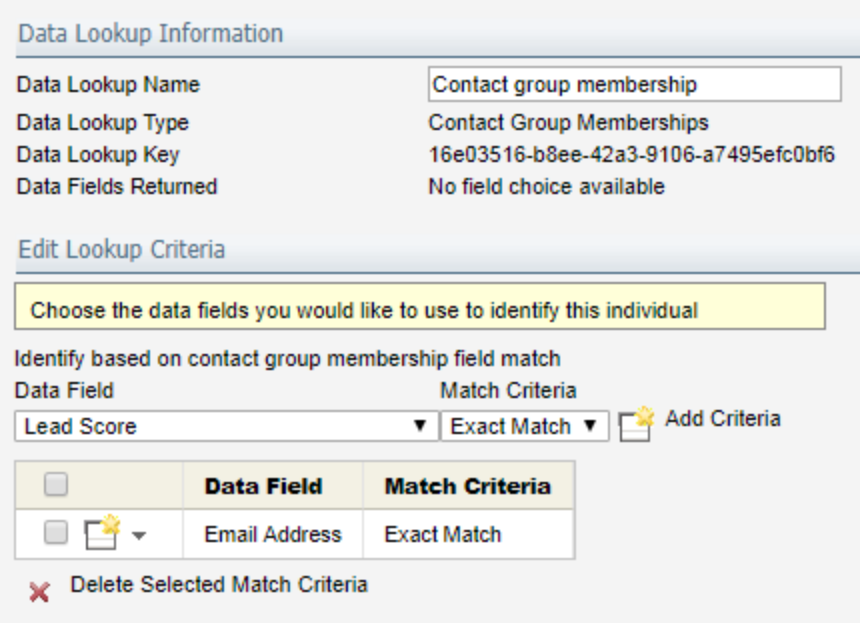
Data lookups allow you to retrieve contact, visitor, or custom object data. Implementing a data lookup on your website allows you to personalize web pages with contact or visitor information.

Creating a custom preferences center

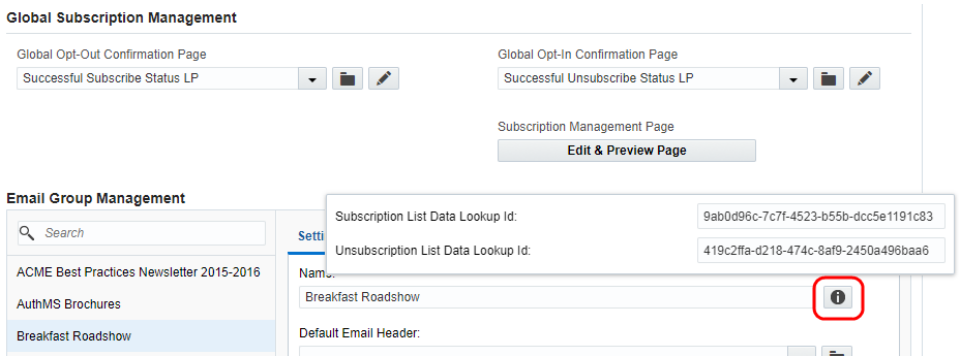
You can use the web data lookup to create a customized email preferences center. In this case, a form is used to show the email group options. When a subscriber follows a subscription management link in their email, the form is pre-populated with their group membership.

To use a web data lookup in a custom preferences center:

1. Create a data lookup that uses Contact Group Memberships. In this case, both the email address and contact ID are used to identify the contact.



2. For each email group, retrieve the subscribe and unsubscribe data lookup IDs. You can get these IDs by opening the email group (**Assets** > **Email Setup** > **Email Groups**) and clicking .



3. Implement the data lookup scripts on your web page. The web page must also have the Oracle Eloqua tracking scripts implemented.

4. When you implement the data lookup, you'll use the group subscribe or unsubscribe lookup IDs.

```
var strContactGroupGUID = '{<group_subscribe_or_unsubscribe_
```

```
lookup_ID>}';
```

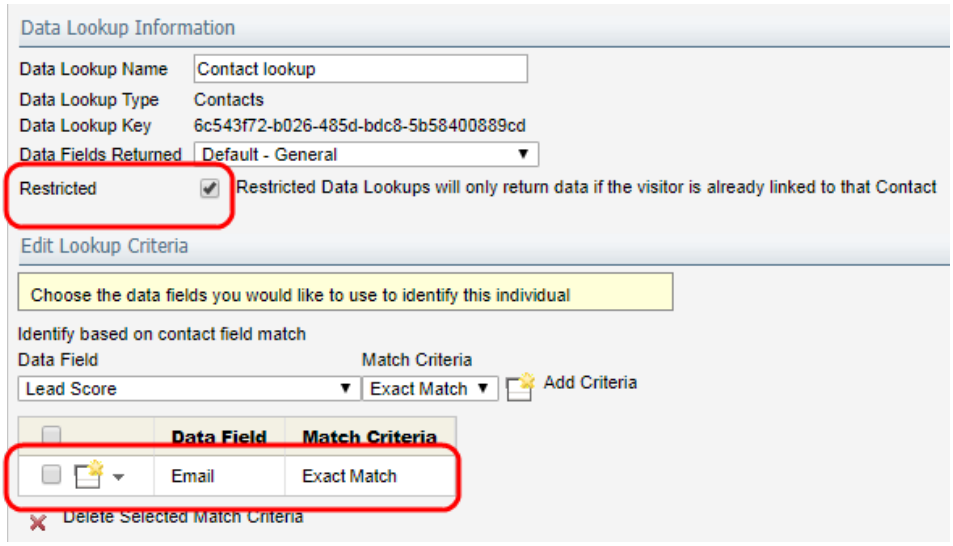
Pre-populating a form

You can use data lookups to pre-populate a form. The form must exist in Oracle Eloqua to use this example

> **Tip**: If you want to pre-populate an Oracle Eloqua form, it is recommend that you use field merges to populate the form field.

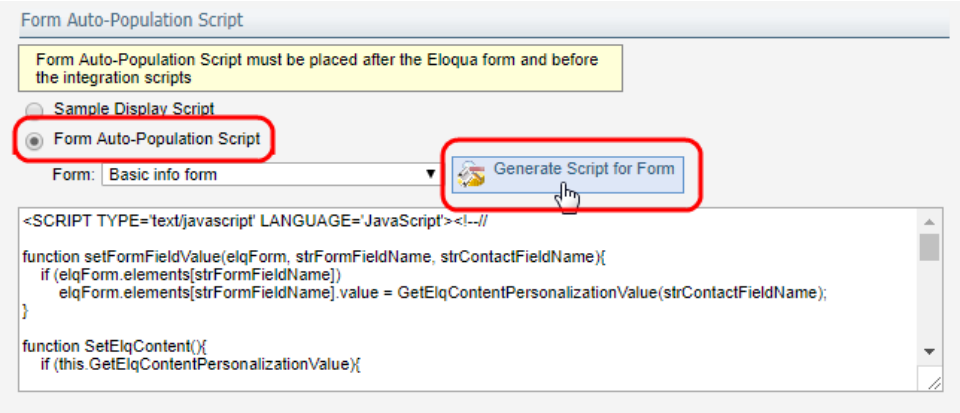
To pre-populate a form with a web data lookup:

1. Create a contact data lookup. In the image below, the contact data lookup uses the email address to identify the contact. The **Restricted** option ensures that contact data is only returned if contact has a visitor profile.



2. Get the sample implementation code by selecting **Data Lookup Options > Get Data Lookup Scripts**.

3. On the *Data Lookup Script* page, choose the **Form Auto-Population Script** option. Choose the form and then click **Generate Script for Form**. This generates sample code specific to selected form.



4. Implement the data lookup on your web page. The web page must also have the Oracle Eloqua tracking scripts implemented.

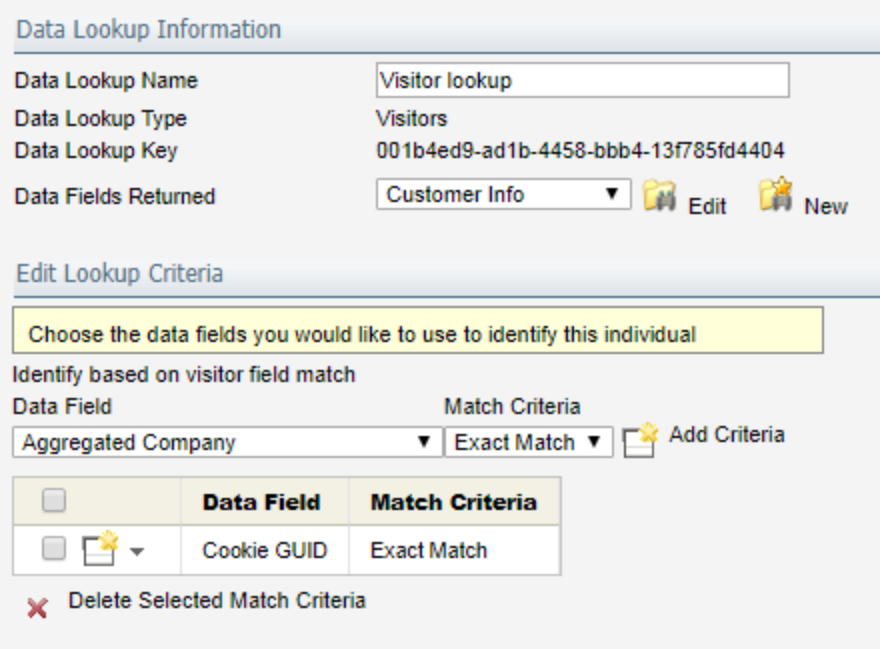
5. When the web page is loaded, the contact's email address must be passed to the page. This could be achieved, for example, by linking to the form from an Oracle Eloqua email.

Personalizing your website using visitor information

You can use a web data lookup to customize your website with visitor information from Oracle Eloqua. You can retrieve any of the fields available from the visitor profile.

To personalize your website with visitor information:

1. Create a visitor data lookup. Visitors are always identified using the Oracle Eloqua cookie global unique identifier (GUID).

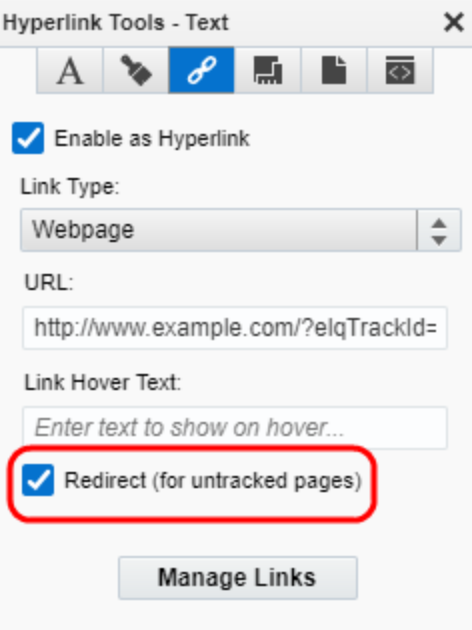


2. Implement the data lookup scripts on your web page. The web page must also have the Oracle Eloqua tracking scripts implemented.
3. After implementing the data lookup scripts, the visitor data lookup will return the visitor profile fields configured in the visitor profile view.

18 Redirect links

A redirect link is an Oracle Eloqua generated URL that allows you to track links to untracked web pages or files. Your organization's website is set up with Oracle Eloqua tracking scripts so that you can track usage. Oracle Eloqua landing pages are also tracked by default. However, if for example, you link to a third-party website from an email, you can track clickthroughs using redirect links.

Marketers can create redirect links when they add a link to an email or landing page and choose the redirect link option:



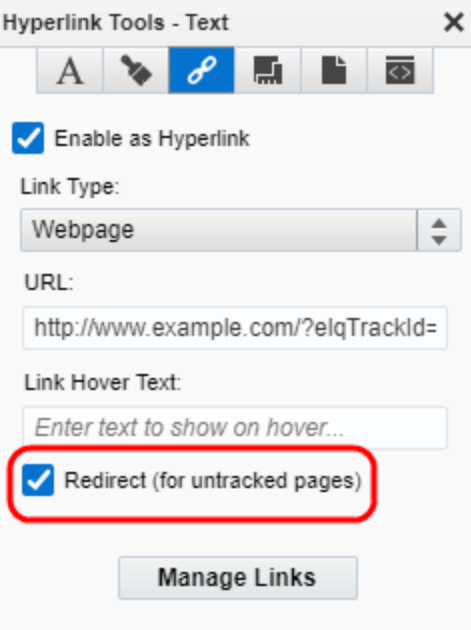
Choosing the redirect option adds the `elqTrack=true` parameter to the URL. After saving the redirect link, Oracle Eloqua generates a unique URL for the redirect. When a contact follows the redirect link, the Oracle Eloqua generated URL link opens first. Then the contact is redirected to the destination URL.

Redirect links are also generated when you add files to the file storage component library.

You can view your organization's the redirect links by navigating to **Assets** > **Website Setup** > **Redirect Links**.

18.1 Changing the destination URL of a redirect link

Marketers can create redirect links when they add a link to an email or landing page and choose the redirect link option:



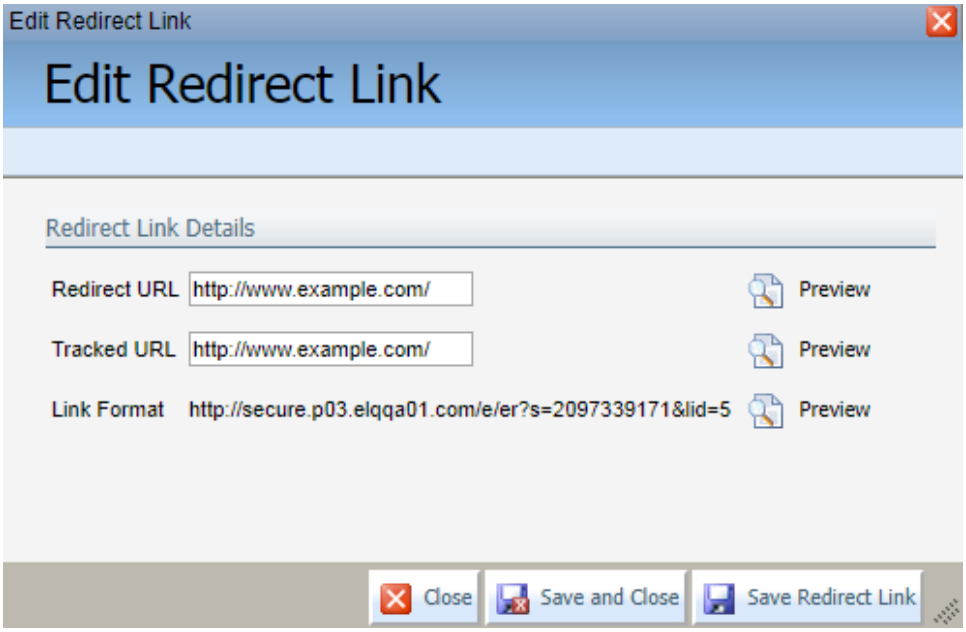
You can change the destination URL of a redirect link even after the link is part of an active landing page or email. This is useful, for example, if you sent out an email with an invalid or incorrect link.

> **Note**: You cannot change the Oracle Eloqua generated URL. You cannot delete a redirect link. Retaining redirect links in Oracle Eloqua allows for tracking history.

To change the destination URL of the redirect link:

1. Navigate to **Assets** > **Website Setup** > **Redirect Links**.
2. Locate the redirect link that you want to change. You can search using wildcards (? for a single character or * for multiple characters).
3. Click the link to open it.

   The *Edit Redirect Link* page appears.



4. Change the URL.
   - To change the link destination, enter a new URL in the **Redirect URL** field. This is where a user will be directed when they follow the link from the email or landing page.
   - To change link as it appears in Oracle Eloqua reports, enter a new value in the **Tracked URL** field. This is only used for reporting purposes. A contact will still be directed to the URL in the Redirect URL field.
5. Click **Preview** to verify the link.
6. Click **Save and Close**.