Auto Event Tracking

Version: 1.0

Author: 6sense

Table Of Contents

Introduction

This guide contains details on implementing our data collection cycle via JavaScript. Its purpose is to collect web requests that roll up activity to the company level. This activity is used in 6sense's models and lets us build strong features around customer intent specific to your products.

The data collection tracks pageloads, events, and other visitor activity. This content is technical in nature, and the intended audiences are teams that typically deploy JavaScript or other tags on the web.

Setup

1. Initialize the 6sense placeholder:

```
window._6si = window._6si || [];
```

2. Event tracking configuration:
    - Interaction with the following elements on a page can be evaluated by our tag:
        - FORM
        - A (anchor tags)
        - BUTTON
        - VIDEO
        - OBJECT
    - To track ALL of the events:

```
window._6si.push(['enableEventTracking', true]);
```

    - To track any combination of events, just include any of the tags above. For example, the following will track anchor tags and buttons only:

```
window._6si.push(['enableEventTracking', true, ['A', 'BUTTON']])
```

    - If you would like to manually track events, then please read our documentation on manual event tracking
3. Set the endpoint for the tag to send events via a 1x1 image pixel:
    - For QA:

```
window._6si.push(['setEndpoint', 'bqa.6sc.co']);
```

    - For production:

```
window._6si.push(['setEndpoint', 'b.6sc.co']);
```

4. Set your token:
    - During onboarding, you should have been given a token to identify your campaign. If you do not have a token, please reach out to your customer success manager.

```
window._6si.push(['setToken', 'CUSTOMER_TOKEN']);
```

5. Load the JavaScript to initialize event tracking:

```
(function() {
      var gd = document.createElement('script');
      gd.type = 'text/javascript';
      gd.async = true;
    gd.src = '//j.6sc.co/6si.min.js';
      var s = document.getElementsByTagName('script')[0];
      s.parentNode.insertBefore(gd, s);
})();
```

Example Script

Please **read carefully** through the configuration steps above before using the configuration below. There are important configurations that must be considered.

```
<script>
  window._6si = window._6si || [];
  window._6si.push(['enableEventTracking', true]);
  window._6si.push(['setToken', 'CUSTOMER_TOKEN']);
  window._6si.push(['setEndpoint', 'b.6sc.co']);

  (function() {
    var gd = document.createElement('script');
    gd.type = 'text/javascript';
    gd.async = true;
    gd.src = '//j.6sc.co/6si.min.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(gd, s);
  })();
</script>
```

Event data sent to the server

On all events captured, we send the following information:

- HTML page "title" value.
- HTML metatag "description" value
- HTML metatag "keywords" value
- HTML custom metatags other than “description", "title" and "keywords”, which you can set using `window._6si.push(['setCustomMetatags',[{},{}]]);`

Page Load

Once the script has initialized, a pageload event is sent to the server. This includes the data sent on every request and any data which is set using `window._6si.push(['setPageAttributes', { attribute: 'value' }]);`. Please read the section on custom configurations to learn more about `setPageAttributes`.

HTML Form

We track the following information:

Form element name and value.

- If a form element value is of type email, then we hash the email before sending it to the server. We also track the domain name from the email address.
- We capture hidden input fields. These fields typically contain metadata relevant to our models. This metadata is typically provided by a marketing automation platform such as Marketo or Eloqua.
- If a form element value contains tabs, line breaks or new lines, then we strip them before sending to our servers.

HTML Anchor

```
<a id="contact_page" href="http://www.6sense.com/contact_us">link</a>
```

On link clicks, we track the href value.

HTML Video

```
<video id="product_video" controls>
    <source src="http://www.6sense.com/videos/nnn" type="video/ogg">
</video>
```

On click, load, play, and stop events:
onclick, onloadeddata ,onplay, and onstop,
we track the video src value.

HTML Button

```
<button id="test"> download</button>
```

On button clicks, we track the button value. In this case, it would be *download*.

HTML Object

```
<div style="width:400px; height:400px;">
    <object id="product_object" width="400" height="400"
data="http://www.6sense.com/product.swf">
        <param name="wmode" value="transparent" />
    </object>
</div>
```

We track all object click events.

Note: To handle click events on <Object>, we add a as a parent element of <Object> and handle the onmousedown event. Also, we add <param name="wmode" value="transparent" /> to enable the mousedown event.

Custom Configurations

Whitelist Fields

Automatically, we capture a hash of any email domain entered in any form. We also include hidden fields, since these usually capture marketing automation metadata that is relevant to our analytics models. If there is any additional information that should be captured and sent to sixsense, please

include it in the whitelist fields.

```
window._6si.push(['setWhitelistFields', ['company_name']]);
```

Page Attributes

Page attributes are commonly used to send additional information on a page or user. For example, if you already have data collected on the current user, you may send that data to us through page attributes. Another common case is to send data on your current campaign (topics, etc.) via these attributes.

To include page attributes, `setPageAttributes` may be used.

```
window._6si.push(['setPageAttributes', {
    campaign_id: 234,
    user_location: 'USA'
}]);
```

The page attributes will be sent to our server only once through the pageload event.

Third Party Analytics

Third party analytics data is typically stored as a cookie on your webpage. In order to add these values and send to us to expand our data models, this portion is relevant.

To include third party values, `setThirdPartyValues` may be used.

```
window._6si.push(['setThirdPartyValues', [
    ['googleAnalytics', '_ga'],['marketo', '_mkto']]
]);
```

The array should contain an array of with two values in it. The first value will represent the name of the vendor. The second value will be the name of the cookie key on your site.

Custom Metatags

If you would like to send additional metatags to the server, other than “description", "title" and "keywords”, then `setCustomMetatags` may be used.

```
window._6si.push(['setCustomMetatags', [
    {
        name: 'details',
    },
    {
        name: 'error',
    },
    {
        name: 'campaign_id',
    },
    {
        name: 'campaign_topics',
    },
]]);
```

```
    <meta name="details" content="product page for users to sign up" />

    <meta name="error" content="no current error" />

    <meta name="campaign_id" content="234">

    <meta name="campaign_topics" content="computer whitepapers and discusion,
software, technology">
```

The custom metatags will be sent to our server on every captured event.

Vimeo Video Integration

Vimeo Integration requires manual configuration.

Embedding videos using the <iframe> player API is the most common and supported way to add Vimeo videos to a website.
A typical Vimeo video will look like this:

```
    <iframe type="text/html" width="640" height="390"
        src="https://player.vimeo.com/video/92866628"
        title="Joseph Puthussery, Cisco and 6sense"
        frameborder="0" allowfullscreen>
    </iframe>
```

To enable tracking of events on the embedded vimeo video, extra parameters must be added to the <iframe> tag:

- Add an id attribute to the <iframe>. This id will be used when pushing events
- Add the class **6si_vimeo_video**. This class will tell the tracking code which videos should be tracked and which ones should not.

After adding these items to the <iframe> tag, it should look like this:

```
    <iframe id="player" class="6si_vimeo_video" width="420" height="315"
        src="https://player.vimeo.com/video/92866628"
        frameborder="0" allowfullscreen>
    </iframe>
```

To track events on an iframe video, you must use the [Vimeo player API (https://github.com/vimeo/player.js)](https://github.com/vimeo/player.js).
The script's onload function can be used to invoke the callback onVimeoIframePlayerReady. In this callback, listeners for the pause and play events can be triggered.

A typical set up would be

```
    /**
    * Vimeo Player API
    * this is boilerplate code to append a new script tag with the Vimeo API
    **/
    var scriptTag = document.createElement('script');
    scriptTag.type = 'text/javascript';
    scriptTag.src = 'https://player.vimeo.com/api/player.js'

    scriptTag.onload = initPlayer

    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(scriptTag, s);

    /**
    * onVimeoIframePlayerReady sets up the pause and play event listeners when
the frame has loaded
    **/
    function onVimeoIframePlayerReady() {
      /**
      * We use jQuery for better compatibility, but you may use native DOM
methods or your tool of choice
      **/

      $('.6si_vimeo_video').each(function() {
          var iframe = $(this)[0];

          var player = new Vimeo.Player(iframe);

          player.on('play', function() {
              window._6si.push(['send', 'play', { event_id: iframe.id,
event_value: iframe.src }]);
          });
          player.on('pause', function() {
              window._6si.push(['send', 'pause', { event_id: iframe.id,
event_value: iframe.src }]);
          });
      });
    }
```

YouTube Integration

YouTube Video Integration requires additional configuration.

Embedding videos using the <iframe> player API is the most common and supported way to add YouTube videos to a website.
A typical YouTube iframe looks like this:

```
    <iframe type="text/html" width="640" height="390"
        src="https://www.youtube.com/watch?v=ntRjkhJ3dPk"
        frameborder="0" allowfullscreen>
    </iframe>
```

To enable tracking of events on the video, extra parameters need to be added to the <iframe> tag:

- Add an id attribute to the <iframe>. This id will be used when pushing events
- Enable the Javascript API by adding the following string to the URL: **?enablejsapi=1**
- Add the class **6si_youtube_video**. This class will tell the tracking code which videos should be tracked and which ones should not.

After adding these items to the <iframe> tag, it should look like this:

```
    <iframe id="player" class="6si_youtube_video" width="420" height="315"
        src="https://www.youtube.com/embed/M7lc1UVf-VE?enablejsapi=1"
        frameborder="0" allowfullscreen>
    </iframe>
```

To track events on an iframe video, the tracking code uses the YouTube iFrame API (https://developers.google.com/youtube/iframe_api_reference) which uses a callback function named onYouTubeIframeAPIReady which may or may not be implemented on your webpage.

A typical set up would be:

```
    /**
    * iFrame API (for iframe videos)
    * this is boilerplate code to append a new script tag with YouTube iFrame
API
    **/
    var tag = document.createElement('script');
    tag.type = 'text/javascript';
    tag.src = "https://www.youtube.com/iframe_api";
    var firstScriptTag = document.getElementsByTagName('script')[0];
    firstScriptTag.parentNode.insertBefore(tag, firstScriptTag);

    /**
    * onYouTubeIframeAPIReady is called for each player when it is ready
    **/
    window.onYouTubeIframeAPIReady = function(){
      /**
      * We use jQuery for better compatibility, but you may use native DOM
methods or your tool of choice
      **/

      $('.6si_youtube_video').each(function() {
          var iframe = $(this);
          var player = new YT.Player(iframe[0], {
              events: {
                  onReady: function(e){
                      e.target._donecheck=true;
                  },
                  onStateChange: function(e){
                      onStateChange(iframe[0], e);
                  }
              }
          });
```

```
      });
    };

    //execute the API calls for play, pause, and finish
    window.onStateChange = function(player, state) {
        if(state.data === 0) {
            onFinish(player.id, player.src);
        } else if(state.data === 1) {
            onPlay(player.id, player.src);
        } else if(state.data === 2) {
            onPause(player.id, player.src);
        }
    };

    //for each of the above three states, make a custom event API call
    window.onPause = function(id, src) {
        window._6si = window._6si || [];
        window._6si.push(['send', 'pause', { event_id: id, event_value: src
}]);
    };

    window.onFinish = function(id, src) {
        window._6si = window._6si || [];
        window._6si.push(['send', 'finish', { event_id: id, event_value: src
}]);
    };

    window.onPlay = function(id, src) {
        window._6si = window._6si || [];
        window._6si.push(['send', 'play', { event_id: id, event_value: src }]);
    };
```